PENENTUAN JUMLAH KENDARAAN MENGGUNAKAN *BLOB DETECTION* DAN *BACKGROUND SUBTRACTION*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
San Sayidul Akdam Augusta
NIM: 145150201111169

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENENTUAN JUMLAH KENDARAAN MENGGUNAKAN *BLOB DETECTION* DAN *BACKGROUND SUBTRACTION*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
San Sayidul Akdam Augusta
NIM: 145150201111169

Skripsi ini telah diuji dan dinyatakan lulus pada
1 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Yuita Arum Sari, S.Kom., M.Kom	Putra Pandu Adikara, S.Kom., M.Kom
NIK: 2016098807152001	NIP: 19850725 200812 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 18 Juli 2018

San Sayidul Akdam Augusta

NIM: 145150201111169

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT karena atas segala rahmat dan limpahan hidayahNya, skripsi yang berjudul "Penentuan Jumlah Kendaraan Menggunakan *Blob Detection* dan *Background Subtraction*" ini dapat disusun dengan baik. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada Program Studi Teknik Informatika, Universitas Brawijaya.

Pada kesempatan ini penulis mengucapkan banyak terima kasih atas segala bantuan dan dedikasi moral maupun material dalam rangka penyusunan skripsi ini. Atas bantuan yang diberikan, penulis mengucapkan banyak terima kasih kepada:

1. Yuita Arum Sari, S.Kom., M.Kom. selaku dosen pembimbing pertama yang membantu dan dengan senang hati menyalurkan ilmu kepada penulis dalam penyusunan skripsi ini.
2. Putra Pandu Adikara, S.Kom., M.Kom. selaku dosen pembimbing kedua yang juga telah memberikan ilmu, bimbingan, arahan, dukungan, serta berbagai macam masukan untuk menyelesaikan skripsi ini.
3. Keluarga besar "SAN" yang telah mendukung penulis dengan segala usahanya, mulai dari doa, materi, dukungan moral, semangat hidup, dan tauladan yang semata-mata untuk keberhasilan penulis.
4. Bapak dan Ibu dosen yang telah mendidik dan memberikan ilmu selama Penulis menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya dan segenap karyawan di Fakultas Ilmu Komputer Universitas Brawijaya yang membantu Penulis dalam pelaksanaan skripsi ini.
5. Teman-teman Program Studi Teknik Informatika yang selalu memberikan semangat, dukungan, dan kebersamaan selama Penulis menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.

Penulis menyadari bahwa skripsi ini tentunya tidak terlepas dari berbagai kekurangan dan kesalahan. Oleh karena itu, segala kritik dan saran yang bersifat membangun sangat penulis harapkan dari berbagai pihak demi penyempurnaan penulisan skripsi ini.

Akhirnya penulis berharap agar skripsi ini dapat memberikan sumbangan dan manfaat bagi semua pihak yang berkepentingan.

Malang, 18 Juli 2018

San Sayidul Akdam Augusta
san.akdam@gmail.com

ABSTRAK

San Sayidul Akdam Augusta, Penentuan Jumlah Kendaraan Menggunakan *Blob Detection* dan *Background Subtraction*

Pembimbing: Yuita Arum Sari, S.Kom., M.Kom dan Putra Pandu Adikara, S.Kom., M.Kom.

Lalu lintas merupakan masalah penting karena lalu lintas adalah sarana untuk berpindah dari suatu tempat ke tempat lain. Apabila lalu lintas terganggu atau terjadi kemacetan maka mobilitas masyarakat juga akan mengalami gangguan. Data kepadatan lalu lintas merupakan peranan penting untuk mengetahui kondisi lalu lintas. Saat ini untuk memperoleh data kepadatan lalu lintas masih dilakukan dengan cara konvensional, yakni dengan menugaskan beberapa orang untuk menghitung setiap kendaraan yang lewat pada rentang waktu tertentu. Tujuan dari penelitian ini dapat menerapkan algoritme *Background Subtraction* dan *Blob* Detection untuk penentuan jumlah kendaraan dan menguji hasil kinerja sistem penentuan jumlah kendaraan. *Background Subtraction* digunakan pada proses segmentasi dengan memisahkan objek dengan *background* dengan menghitung selisih tiap piksel dan memanfaatkan *threshold* untuk menjadikan dua grup piksel yang dominan. Metode yang digunakan untuk menentukan posisi objek dan jumlah kendaraan menggunakan *Blob Detection* dan *Background Subtraction*. Pengujian dilakukan dengan menggunakan dua puluh citra dengan mengambil nilai *error* terkecil sebagai hasil evaluasi terbaik. Kinerja yang dihasilkan adalah *precision* sebesar 93,44%, *recall* sebesar 77,03% dan akurasi sebesar 73,75%. Maka nilai *precision*, *recall* dan akurasi perlu ditingkatkan lagi dengan menambahkan parameter pengujian dan memperbanyak *dataset* dengan kondisi yang berbeda-beda. Hasil menunjukkan bahwa metode *Blob Detection* dan *Background Subtraction* dapat memberikan hasil yang cukup baik apabila *blob* antar kendaraan berjarak. Metode ini tidak memberikan hasil yang baik apabila digunakan pada kondisi lalu lintas yang padat dengan badan kendaraan saling menempel.

Kata kunci: *Background Subtraction*, *Blob Detection, object filtering,* deteksi objek, *preprocessing*

ABSTRACT

San Sayidul Akdam Augusta, Penentuan Jumlah Kendaraan Menggunakan *Blob Detection* dan *Background Subtraction*

***Adviser*: Yuita Arum Sari, S.Kom., M.Kom *and* Putra Pandu Adikara, S.Kom., M.Kom.**

Traffic is an important problem because traffic is medium to move from one place to another. When there is a traffic problem or become stuck, then people's mobility will have problem too. Traffic density data is an important role to understand traffic condition. Currently, in order to obtain traffic density still do it in conventional way, that is with some people to count each vehicles passing by at certain time. The purpose of this research is to apply algorithm Background Subtraction dan Blob Detection to determine total vehicles and test the result of the vehicle counter system. Background Subtraction is used to process segmentation to separate an object with the background by counting difference between each pixel and use a threshold to make two dominant group of pixel. The method used to determine object position and total vehicles by Blob Detection and Background Subtraction. Testing done with twenty image by taking smallest error value as a best evaluation. The performance of precision is 93.44%, recall is 77.03% and accuracy is 73.75%. The value of precision, recall and accuracy needs to be increased again by adding test parameters and multiplying datasets with different conditions. The results show that the Blob Detection and Background Subtraction methods can give pretty good results when blob between vehicles is spaced. This method does not provide good results when used in heavy traffic conditions with vehicle bodies sticking together.

Keywords: Background Subtraction, Blob Detection, object filtering, object detection, preprocessing

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Pada bab ini dijelaskan tentang masalah yang ada atau latar belakang yang mendasari untuk pembuatan sistem penentuan jumlah kendaraan menggunakan *Blob Detection* dan *Background Subtraction*, serta beberapa teori yang akan mendukung dan berhubungan dengan penelitian yang akan dibuat.

1.1 Latar belakang

Lalu lintas merupakan masalah penting karena lalu lintas adalah sarana untuk berpindah dari suatu tempat ke tempat lain. Apabila lalu lintas terganggu atau terjadi kemacetan maka mobilitas masyarakat juga akan mengalami gangguan (Mediawati, 2017). Gangguan ini dapat menyebabkan pemborosan bahan bakar, pemborosan waktu dan dapat mengakibatkan polusi udara. Kemacetan lalu lintas umumnya dapat terjadi karena ketidakmampuan suatu jalan dalam melayani arus kedatangan atau jumlah arus kedatangan berlebih dan tidak teratur dalam satuan waktu tertentu dan dalam ruas jalan tertentu maupun disebabkan oleh keduanya.

Pakar lalu lintas senior Tom Tom Nick Cohn menempatkan Jakarta menempati posisi 4 dalam hal kota dengan kemacetan lalu lintas terburuk pada jam sibuk di dunia dan menunjukkan tingkat kemacetan di ibu kota Indonesia ini adalah 58% dengan puncaknya ketika pagi mencapai 63% (Pitoko, 2017). Data kepadatan lalu lintas merupakan peranan penting untuk mengetahui kondisi lalu lintas. Saat ini untuk memperoleh data kepadatan lalu lintas masih dilakukan dengan cara konvensional, yakni dengan menugaskan beberapa orang untuk menghitung setiap kendaraan yang lewat pada rentang waktu tertentu (Cahyana, 2014). Penugasan tersebut dimungkinkan terjadi *human error* dalam proses perhitungan diakibatkan terlalu padatnya kendaraan, pengaruh lingkungan atau tidak semua kendaraan terjangkau oleh mata. Mengakibatkan berkurangnya keakuratan dalam proses perhitungan. Selain rentan terjadinya *human error,* perhitungan yang dilakukan manusia mempunyai biaya tersendiri untuk setiap penugasannya sehingga kurang efisien.

Dari permasalahan di atas, penentuan jumlah kendaraan memiliki peran penting dengan memanfaatkan citra untuk mengetahui jumlah kendaraan yang lewat pada saat itu. Penelitian sebelumnya sistem pendeteksi jumlah mobil dalam *Intelligent Transport System* (ITS) menggunakan metode *Viola-Jones* mempunyai akurasi sebesar 92,86% pada kondisi sampel sebuah mobil, keakuratan terendah sebesar 52,78% pada kondisi sampel acak, sehingga *Viola-Jones* sangat baik dalam mendeteksi mobil dalam kondisi sampel sebuah mobil (Harwendhani, 2016). Pada penelitian lain menggunakan metode *Background Subtraction* untuk *tracking* dan klasifikasi kendaraan dilakukan pada siang, sore dan malam hari, mendapatkan akurasi rata-rata sebesar 96% pada *threshold* 30 dan akurasi rata-rata terendah 86% pada *threshold* 20 (Samir, et al., 2016). Metode *Blob Detection* dapat digunakan pada deteksi kendaraan dengan kinerja tingkat sensitivitas sebesar 91,67%, presisi sebesar 61,11%, kekhususunan sebesar 80,55%, *f-Measure* sebesar 73,33% dan akurasi sebesar 83,33%.

Walaupun nilai presisi lebih rendah dari pada nilai sensitivitas, metode yang digunakan sudah dapat dikatakan efektif karena nilainya di atas 50% (Hidayati, 2017). Penelitian terakhir *Blob Detection* diterapkan untuk mengetahui kerusakan pada permukaan kertas dengan menggabungakan *median filtering* untuk menemukan lubang permukaan, titik terang, titik gelap yang muncul pada kertas lebih akurat yang mana melebihi 89%. Membandingkan dengan algoritme sebelumnya dalam mendeteksi kerusakan pada kertas, algoritme *Blob Detection* lebih cepat, *real time*, efisien, posisi yang akurat dan diimplemetasikan pada sistem yang besar (Qi, et al., 2013).

Berdasarkan permasalahan dan alasan yang telah dipaparkan di atas dan didukung oleh beberapa penelitian sebelumnya dilakukan maka pada penelitian ini akan dibangun sebuah sistem untuk menentukan jumlah kendaraan menggunakan gabungan dari *Background Subtraction* sebagai algoritme pemisah objek dan *background* pada citra dan *Blob Dection* sebagai algoritme untuk menghitung jumlah kendaraan dari hasil pemisahan antara objek dan *background.* Dengan demikian sistem yang akan dibangun ini diharapkan dapat membantu seoptimal mungkin terhadap permasalahan untuk menentukan jumlah kendaraan dengan melalui pemisahan antara objek dan *background* terlebih dahulu.

1.2 Rumusan masalah

Berdasarkan latar belakang di atas tentang permasalahan penentuan jumlah kendaraan menggunakan *Blob Detection* dan *Background Subtraction* maka didapatkan rumusan masalah sebagai berikut:

1. Bagaimana cara menerapkan penggabungan algoritme *Background Subtraction* dan *Blob Detection* untuk permasalahan penentuan jumlah kendaraan pada citra?
2. Bagaimana hasil kinerja yang diperoleh dari penerapan penggabungan algoritme *Background Subtraction* dan *Blob Detection* untuk permasalahan penentuan jumlah kendaraan pada citra?

1.3 Tujuan

Tujuan dari perancangan sistem penentuan jumlah kendaraan adalah sebagai berikut:

1. Menerapkan algoritme *Background Subtraction* dan *Blob Detection* untuk penentuan jumlah kendaraan.
2. Menguji hasil kinerja sistem penentuan jumlah kendaraan yang menggunakan algoritme *Background Subtraction* dan *Blob Detection.*

1.4 Manfaat

Manfaat dari perancangan sistem penentuan jumlah kendaraan maka didapatkan manfaat sebagai berikut:

1. Bagi Penulis
    a. Dengan pembuatan sistem ini penulis mendapatkan ilmu untuk menerapkan pengetahuan baik yang sudah didapatkan selama

melakukan perkuliahan maupun pengalaman-pengalaman baru yang sebelumnya belum pernah didapatkan.

b. Dengan pembuatan sistem ini penulis mendapatkan ilmu dalam memahami cara mengimplementasikan algoritme *Background Subtraction* dan *Blob Detection* untuk penentuan jumlah kendaraan pada citra.

2. Bagi Pengguna
   a. Sistem ini dapat memudahkan dalam mendapatkan informasi kepadatan lalu lintas dengan mengetahui jumlah kendaraan secara otomatis pada citra pada saat tertentu.

1.5 Batasan masalah

Berdasarkan penjelasan-penjelasan sebelumnya, ruang lingkup atau batasan yang digunakan dalam perancangan sistem penentuan jumlah kendaraan kemacetan lalu lintas adalah sebagai berikut:

1. Data yang digunakan untuk penentuan jumlah kendaraan adalah gambar yang diambil dari beberapa *video* dari *Youtube*, detail informasi terletak pada subbab 3.4.
2. Data yang digunakan disimpan ke dalam format jpg untuk digunakan secara bertahap ke dalam algoritme.
3. Metode yang digunakan dalam menghitung jumlah kendaraan adalah *Background Subtraction* dan *Blob Detection*.

1.6 Sistematika pembahasan

Pada subbab sistematika pembahasan ini digunakan untuk memberikan secara umum gambaran tentang apa saja yang nantinya akan dibahas pada sistem penelitian ini, sehingga secara sekilas pembaca akan mengetahui poin-poin penting dalam penelitian ini. Berikut sistematika dari penelitian ini:

BAB I PENDAHULUAN

Pada bab ini sepertinya pada umumnya yang pertama adalah latar belakang, kedua membahas tentang rumusan masalah, ketiga membahas tentang tujuan dibuatnya penelitian ini, keempat membahas batasan-batasan yang menjadi tujuan sistem dan terakhir adalah sistematika pembahasan yang digunakan sebagai poin umum agar pembaca mengetahui poin-poin penting dalam penelitian ini.

BAB II LANDASAN KEPUSTAKAAN

Pada bab ini berisi teori yang sudah ada atau dari penelitian yang sudah pernah dilakukan dengan tujuan untuk mendukung sekaligus menjadi pondasi sebuah sistem yang akan dibangun. Dasar teori sangat diperlukan demi tercapainya suatu tujuan sistem yang akan dibangun. Dasar teori yang akan digunakan pada

penelitian ini adalah ulasan kepadatan lalu lintas, *image processing,* algoritme *Background Subtraction* dan *Blob Detection.*

BAB III METODE PENELITIAN

Pada bab ini terdapat langkah-langkah paling dasar yang akan digunakan dalam membangun penelitian ini, yang pertama adalah studi pustaka yakni dengan mempelajari teori dan penelitian terdahulu, kedua adalah analisis kebutuhan dimana kita harus tau apa saja yang akan sistem butuhkan dan pengguna butuhkan, ketiga adalah pengumpulan data yang digunakan sebagai bahas dalam pengujian, keempat adalah pengolahan data, keenam adalah perancangan sistem, ketujuh adalah implementasi, yang terakhir adalah pengujian, dan pengambilan kesimpulan.

BAB IV PERANCANGAN SISTEM

Pada bab ini berisi perancangan sistem secara detail yang digunakan dalam mengembangkan sistem untuk analisis sentimen dan perbaikan kata yang didasarkan dengan teori dan penelitian sebelumnya yang sudah dipelajari.

BAB V IMPLEMENTASI SISTEM

Bab ini menjelaskan tentang penerapan algoritme Background Subtraction dan Blob Detection dalam penentuan jumlah kendaraan. Pada implementasi yang akan dibangun oleh sistem harus didasarkan dengan analisa dan perancangan yang sebelumnya sudah dilakukan dengan benar menurut teori yang digunakan.

BAB VI PENGUJIAN DAN ANALISIS

Pada bab ini berisi hasil dari pengujian yang telah dilakukan kepada sistem sesuai dengan prosedur yang digunakan. Pengujian yang paling utama dan harus dilakukan adalah pengujian accuracy yang dihitung dari data uji dan parameter uji yang berbeda-beda. Hasil pengujian berupa perbandingan accuracy antara penentuan jumlah kendaraan oleh sistem dan jumlah kendaraan yang seharusnya didapatkan.

BAB VII PENUTUP

Pada bab ini berisi kesimpulan dari pembuatan sistem visi komputer yaitu penentuan jumlah kendaraan dan juga terdapat saran yang diharapkan dapat menjadi masukan untuk menjadikan sistem ini lebih baik lagi untuk ke depannya.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi uraian dan pembahasan tentang teori, konsep, model, metode, atau sistem dari studi kepustakaan ilmiah, yang berkaitan dengan tema, masalah, atau pertanyaan penelitian. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian. Jika dibutuhkan sesuai dengan karakteristik penelitiannya dan syarat kecukupan khusus keminatan tertentu, bisa juga terdapat kajian pustaka yang menjelaskan secara umum penelitian-penelitian terdahulu yang berhubungan dengan topik skripsi dan menunjukkan persamaan dan perbedaan skripsi tersebut terhadap penelitian terdahulu yang dituliskan.

2.1 Kajian Pustaka

Berdasarkan penelitian yang sudah ada, penulis menggunakan beberapa penelitian yang merujuk kepada penelitian yang hampir sama untuk mendukung penelitian ini. Penelitian pertama dilakukan oleh Irmaya Citra Harwendhani pada tahun 2016 tentang sistem pendeteksi jumlah mobil dalam *Intelligent Transport System* (ITS) menggunakan algoritme *Viola-Jones*. Penelitian kedua dilakukan oleh Muhammad Ikhsan Samir, dkk tentang penerapan algoritme *Background Subtraction* untuk *tracking* dan klasifikasi kendaraan. Penelitian ketiga dilakukan oleh Qory Hidayati tentang kendali lampu lalu lintas dengan deteksi Kendaraan menggunakan algoritme *Blob Detection.*

Pada penelitian yang dilakukan Irmaya Citra Harwendhani menyimpulkan bahwa keakuratan tertinggi algoritme *Viola-Jones* dalam menentukan jumlah mobil sebesar 92,86% yaitu terdapat pada kondisi sampel sebuah mobil. Sedangkan keakuratan terendah sebesar 52,78% diperoleh pada kondisi sampel acak. Hal ini menunjukkan menggunakan algoritme *Viola-Jones* sangat baik jika dalam mendeteksi jumlah mobil dalam kondisi sampel sebuah mobil (Harwendhani, 2016). Berikutnya, penelitian dari Muhammad Ikhsan Samir, dkk yaitu penerapan algoritme *Background Subtraction* untuk *tracking* dan klasifikasi kendaraan dalam penggunaan *threshold* dan kondisi cuaca berpengaruh terhadap pengujian. *Threshold* yang digunakan untuk pengujian yaitu *threshold* 20, 30 dan 40. Hasil rata-rata dari pengujian *threshold* pada waktu pagi, siang dan sore, dengan presentase 86% pada *threshold* 20, 96% pada *threshold* 30 dan 86% pada nilai *threshold* 40. Hal ini menunjukkan bahwa *threshold* optimal dari pengujian dengan menggunakan *threshold* 30 (Samir, et al., 2016). Penelitian selanjutnya dilakukan oleh Qory Hidayati menyimpulkan bahwa tingkat akurasi dalam kondisi cerah sebesar 82,11% berkurang menjadi 76,50% pada kondisi hujan. Akurasi berkurang dalam kondisi ramai disebabkan karena posisinya yang berdekatan, sehingga sistem membaca banyak kendaraan tersebut sebagai satu kesatuan objek kendaraan dengan dimensi yang besar (Hidayati, 2017). *Blob Detection* tidak hanya digunakan dalam pendeteksian kendaraan, penelitian terakhir ini dilakukan oleh Xingguang Qi, dkk menggunakan algoritme untuk

mengetahui kerusakan pada permukaan kertas dengan menggabungakan *median filtering* untuk menemukan lubang permukaan, titik terang, titik gelap yang muncul pada kertas lebih akurat yang mana melebihi 89%. Membandingkan dengan algoritme sebelumnya dalam mendeteksi kerusakan pada kertas, algoritme *Blob Detection* lebih cepat, *real time*, efisien, posisi yang akurat dan diimplemetasikan pada sistem yang besar (Qi, et al., 2013).

Perbedaan pada penelitian ini dengan penelitian sebelumnya adalah menggabungkan algoritme *Background Subtraction* sebagai pemisah objek dengan *background* dan *Blob Detection* digunakan untuk mendeteksi objek dan menghitung objek sebanyak objek yang dideteksi. Sistem yang dibuat akan memberikan hasil untuk mengetahui perbandingan nilai akurasi dengan penelitian sebelumnya. Perbandingan objek dan metode penelitian yang telah dilakukan terhadap tinjauan pustaka dari penelitian sebelumnya ditunjukkan pada Tabel 2.1.

Tabel 2.1 Kajian Pustaka

	Penulis	Objek	Metode	Hasil
1	(Harwendhani, 2016)	Pendeteksi jumlah mobil	Algoritme yang dipakai *Viola-Jones* : a. Proses konversi gambar dari RGB ke *Grayscale.* b. *Haar Cascade Clasifier.* c. *AdaBost.*	Penggunaan *Viola-Jones* keakuratan pendeteksian sistem sangat baik yaitu 92,86% jika kondisi sampel dengan jumlah mobil sedikit yaitu satu mobil dibandingkan pada kondisi sampel pengujian secara acak yaitu 52,78%.
2	(Samir, et al., 2016)	*Tracking* dan klasifikasi kendaraan	Algoritme yang dipakai *Background Subtraction* : a. *Prepocessing.* b. *Background Subtraction.* c. *Thresholding.* d. Klasifikasi.	Hasil rata-rata dari pengujian *threshold* pada waktu pagi, siang dan sore, dengan presentase 86% pada nilai *threshold* 20, 96% pada nilai *threshold* 30 dan 86% pada nilai *threshold* 40. Hal ini menunjukkan *threshold* 30 merupakan hasil optimal dari hasil pengujian.
3	(Hidayati, 2017)	Deteksi kendaraan untuk mengendalikan lampu lalu	Algoritme yang dipakai *Blob Detection*: a. *Prepocessing*. b. Segmentasi	Kesalahan dalam kondisi ramai lebih banyak disebabkan karena posisinya yang bergerombol dan

		lintas	citra c. *Object filtering.* d. Deteksi kendaraan dan perhitungan	berdekatan satu sama yang lain, sehingga sistem membaca banyak kendaraan sebagai satu kesatuan kendaraan dengan dimensi yang besar.
4	(Qi, et al., 2013)	Deteksi kerusakan pada kertas	Blob Detection: a. *Prepocessing*. b. Segmentasi citra *c. Connectivity analysis* *d. Feature extraction* *e. Blob geometric characteristics description*	Menemukan lubang permukaan, titik terang, titik gelap yang muncul pada kertas lebih akurat yang mana melebihi 89%.

2.2 *Prepocessing*

Preprocessing adalah tahap awal sebelum dilakukan pendeteksian. Pada tahap ini bertujuan untuk mempersiapkan komponen-komponen yang akan digunakan seperti inisialisasi. Komponen yang digunakan sebagai input yaitu citra berwarna yang terdiri dari 3 *channel* atau biasa yang disebut dengan citra RGB (*red, green, blue*).

2.2.1 *Grayscale*

Merupakan citra satu kanal pada posisi piksel *(x, y)* yang menyatakan tingkat keabuan dari hitam ke putih, *x* menyatakan variabel baris dan *y* menyatakan variabel kolom. *grayscale* dengan 256 level artinya mempunyai skala abu-abu dari 0 sampai 255 dalam intensitas 0 menyatakan warna hitam dan 255 menyatakan warna putih (Samir, et al., 2016). Proses konversi citra berwarna menjadi *grayscale* menggunakan Persamaan 2.1 (Harwendhani, 2016).

$$X = (0{,}299 * R) + (0{,}587 * G) + (0{,}114 * B) \quad (2.1)$$

Keterangan:
X : derajat keabuan
R : nilai piksel *channel red*
G : nilai piksel *channel green*
B : nilai piksel *channel blue*

Algoritme ini merupakan tahap *preprocessing* digunakan untuk mempermudah tahap pemisahan objek kendaraan dengan *background.*

2.3 Segmentasi

Segmentasi merupakan proses pemisahan objek dengan yang lainnya. Pada proses ini bertujuan untuk memisahkan objek dengan *background*.

2.3.1 *Background Subtraction*

Background Subtraction banyak digunakan pada proyek-proyek berbasis pengolahan citra. Salah satu yang sering menggunakan fungsi dari *Background Subtraction* ini adalah aplikasi penghitung jumlah pengunjung yang memasuki maupun yang meninggalkan ruangan ataupun kendaraan yang melewati suatu jalur dalam sistem informasi lalu lintas (Kurniawan, 2015). Ide dasar dari *Background Subtraction* adalah |*frame(n) – background*| > *threshold,* bila piksel ke *n* memenuhi persamaan tersebut maka piksel tersebut tersebut digolongkan piksel objek dan selain itu merupakan *background* seperti pada Persamaan 2.2 (Samir, et al., 2016).

$$S(n) = |F(n) - B| > T \tag{2.2}$$

Keterangan:
$S(n)$: selisih *frame* ke-*n* dari *video* masukan
$F(n)$: *frame* ke-*n* dari *video* masukan
B : citra *background*
T : *threshold*

Dalam penelitian ini *Background Subtraction* digunakan untuk melakukan pemisahan objek dan *background* dari 2 buah citra, dimana citra yang pertama merupakan citra yang digunakan sebagai *background* dan citra yang kedua merupakan citra acak untuk membaca selisih tiap piksel. Kemudian menjadikan citra keabuan menjadi dua grup dominan atau citra *biner* dengan memanfaatkan nilai *threshold* yang ditentukan, sehingga dapat diketahui daerah mana yang termasuk objek dan daerah mana yang termasuk *background*.

2.4 *Object Filtering*

Object filtering merupakan metode untuk menonjolkan suatu objek pada citra sehingga lebih mudah dibedakan dengan objek lainnya. Pada proses ini bertujuan untuk mengurangi derau.

2.4.1 *Erosion*

Merupakan operasi *morphologi* yang akan mengurangi piksel pada batas antar objek dalam citra dari hasil nilai terkecil himpunan piksel tetangganya (Wirayuda, 2006).

Proses ini merupakan perbaikan citra yang memungkinkan menghilangkan derau yang bukan objek dengan memanfaatkan nilai piksel tetangganya. *Erosion* dapat dihitung menggunakan Persamaan 2.3 (Menegaz , 2017).

$$A \ominus B = \{z|(B)_z \subseteq A\} \tag{2.3}$$

Keterangan:
A : citra masukan
B : citra *structuring element*
z : nilai piksel bagian dari citra A

2.4.2 *Dilation*

Merupakan operasi *morphologi* yang akan menambahkan piksel pada batas antar objek dalam citra dari hasil nilai maksimal himpunan piksel tetangganya (Wirayuda, 2006).

Proses ini merupakan perbaikan citra yang memungkinkan meperjelas objek dari hasil proses segmentasi dengan memanfaatkan nilai piksel tetangganya. *Dilation* dapat dihitung menggunakan Persamaan 2.4 (Menegaz , 2017).

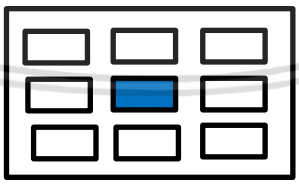
$$A \oplus B = \{z|(B)_z \cap A \neq \emptyset\} \quad (2.4)$$

Keterangan:
A : citra masukan
B : citra *structuring element*
z : nilai piksel bagian dari citra A

2.5 *Blob Detection*

Blob Detection adalah algoritme untuk menentukan suatu grup dari piksel yang saling berhubungan atau tidak. Algoritme ini biasanya digunakan untuk mengidentifikasi objek yang terpisah atau menghitung jumlah objek dari suatu citra yang harus dilakukan algoritme *threshold* terlebih dahulu (Hidayati, 2017).

Piksel bertetangga pada sebuah piksel ditentukan sebagai piksel yang berjarak 1 piksel dari piksel asal. Proses perhitungan *blob* akan memanfaatkan relasi piksel *8-neighbors.* Relasi piksel *8-neighbors* dapat dilihat pada Gambar 2.1 dan keterangan relasi piksel *8-neighbors* dapat dilihat pada Tabel 2.2 (H, et al., 2015).



Gambar 2.1 Relasi *8-neighbors*

Tabel 2.2 Relasi *8-neighbors*

$P(x-1, y-1)$	$P(x, y-1)$	$P(x+1, y-1)$
$P(x-1, y)$	$P(x, y)$	$P(x+1, y)$
$P(x-1, y+1)$	$P(x, y+1)$	$P(x+1, y+1)$

Algoritme ini digunakan untuk mengetahui jumlah objek dalam suatu citra. Proses pemetaan suatu objek akan menelusuri tiap piksel pada baris yang ada

dan memberikan label pada piksel tetangga yang bernilai 255. Satu *blob* merepresentasikan satu objek dan menentukan batas terkecil dan maksimal setiap objek untuk mengurangi kesalahan mendeteksi objek yang berdekatan.

2.6 Deteksi Objek (Kendaraan)

Deteksi objek merupakan proses untuk mengetahui koordinat atau posisi objek sebelum dilakukan perhitungan jumlah objek pada suatu citra.

2.6.1 *Multiple Blob*

Setiap *blob* merepresentasikan suatu objek yang mana tiap objek adalah sekelompok piksel yang mempunyai fitur penting atau label sebagai pembeda dengan objek lainnya. Proses pencarian label untuk sebuah dalam suatu citra dilakukan penelusuran dari atas sampai bawah. Pertama untuk label objek yang bernilai 1. Ketika piksel bernilai 1, mencatat koordinat batas atas atau batas terkecil koordinat sumbu Y dari sebuah objek, kemudian penelusuran dari kiri ke kanan untuk mencatat koordinat batas kiri atau batas terkecil koordinat sumbu X dari sebuah objek, penulusuran dari kanan ke kiri sebagai batas kanan atau batas terbesar koordinat sumbu X dari sebuah objek dan dari bawah ke atas sebagai batas bawah atau batas terbesar koordinat sumbu Y dari sebuah objek. Kemudian dilakukan pencarian lagi untuk label objek yang berbeda. Batas daerah sebuah objek dapat ditujukkan pada Persamaan 2.5 (Qi, et al., 2013; Qi, et al., 2013).

$$
\begin{aligned}
top &= \min y\{(x,y)|(x,y) \in Ri(x,y)\} \\
bottom &= \max y\{(x,y)|(x,y) \in Ri(x,y)\} \\
left &= \min x\{(x,y)|(x,y) \in Ri(x,y)\} \\
right &= \max \mathrm{x}\{(x,y)|(x,y) \in Ri(x,y)\}
\end{aligned} \tag{2.5}
$$

Keterangan:
(x,y) : koordinat sebuah citra
$R(x,y)$: daerah sebuah objek
i : nilai yang menunjukkan daerah sebuah objek
top : batas atas atau batas terkecil koordinat sumbu Y
$bottom$: batas bawah atau batas terbesar koordinat sumbu Y
$left$: batas kiri atau batas terkecil koordinat sumbu X
$right$: batas kanan atau batas terbesar koordinat sumbu X

Algoritme ini digunakan memberikan tanda daerah dalam sebuah objek. Kemudian setiap batas atas, bawah, kanan dan kiri disimpan ke dalam sebuah variabel untuk dilakukan penandaan setiap objek dalam sebuah citra dan dilakukan proses penentuan jumlah kendaraan sesuai banyak daerah atau objek dalam suatu citra.

2.7 Evaluasi

Evaluasi merupakan proses untuk mempermudah dalam mengambil kesimpulan. Dalam penelitian ini dilakukan 2 evaluasi, pertama evaluasi untuk

membandingkan perbedaan jumlah objek yang seharusnya didapatkan dengan jumlah objek yang didapatkan oleh sistem dalam suatu citra dan evaluasi kesalahan mendeteksi objek dalam suatu citra.

2.7.1 *Mean Square Error* (MSE)

MSE adalah sebuah algoritme yang digunakan untuk membandingkan antara nilai yang terkontaminasi dengan kesalahan dan nilai yang seharusnya dengan memberikan skor kuantitatif yang menggambarkan tingkat kemiripan atau tingkat kesalahan diantara keduanya. MSE dihitung dengan Persamaan 2.6 (Wang & Bovik, 2009).

$$MSE(x, y) = \frac{1}{N} \sum_{i=1}^{N} (xi - yi)^2 \quad (2.6)$$

Keterangan:
N : jumlah *dataset*
(x, y) : perbedaan antara nilai x dan y pada *dataset* tertentu
i : nilai yang menunjukkan sebuah percobaan
xi : nilai dihasilkan oleh sistem yang terkontaminasi dengan kesalahan
yi : nilai yang seharusnya dicapai oleh sistem

Pada proses ini nilai error yang dikatakan baik jika *error* mempunyai nilai yang kecil.

2.7.2 *Confusion Matrix*

Confusion Matrix biasanya digunakan untuk mendeskripsikan evaluasi dari klasifikasi. Beberapa metode yang diturunkan dari *confusion matrix* seperti akurasi, *error rate*, *false positive rate, recall*, *specificity* dan *precision*. Tabel 2.3 menunjukkan gambaran *confusion matrix* (Adinugroho & Sari, 2018).

Tabel 2.3 Tabel *Confusion Matrix*

		Nilai yang sebenarnya	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	***TP*** (*True Positive*)	***FP*** (*False Positive*)
	FALSE	***FN*** (*False Negative*)	***TN*** (*True Negative*)

TP : hasil prediksi benar dan nilai sebenarnya benar
TN : hasil prediksi salah dan nilai sebenarnya salah
FP : hasil prediksi benar dan nilai sebenarnya salah
FN : hasil prediksi salah dan nilai sebenarnya benar

2.7.2.1 *Precision*

Precision dihitung dari jumlah keseluruhan nilai prediksi positif yang benar dibagi dengan jumlah keseluruhan prediksi kelas yang benar. Nilai *precision* terbaik jika nilai akurasi tersebut sama dengan 1 dan paling buruk adalah 0. *Precision* dapat dihitung menggunakan Persamaan 2.7 (Adinugroho & Sari, 2018).

$$Precision = \frac{TP}{TP+FP} \quad (2.7)$$

Keterangan:
TP : nilai prediksi benar dan nilai sebenarnya benar
FP : nilai prediksi benar dan nilai sebenarnya salah

Pada penelitian ini *precision* digunakan untuk menghitung persentase dari jumlah kendaraan yang mendeteksi hasil positif dari total jumlah kendaraan yang ditemukan sistem.

2.7.2.2 *Recall*

Recall dihitung dari jumlah prediksi positif yang benar dibagi dengan jumlah keseluruhan kelas yang positif. Nilai *recall* terbaik jika nilai akurasi tersebut sama dengan 1 dan paling buruk adalah 0. *Recall* dapat dihitung menggunakan Persamaan 2.8 (Adinugroho & Sari, 2018).

$$Recall = \frac{TP}{TP+FN} \quad (2.8)$$

Keterangan:
TP : nilai prediksi benar dan nilai sebenarnya benar
FN : nilai prediksi salah dan nilai sebenarnya benar

Pada penelitian ini *recall* digunakan untuk menghitung presentase dari semua kendaraan yang mendeteksi nilai positif.

2.7.2.3 Akurasi

Akurasi merupakan hasil perhitungan semua nilai prediksi yang benar dibagi dengan keseluruhan data. Nilai akurasi terbaik jika nilai akurasi tersebut sama dengan 1 dan paling buruk adalah 0.

Pada penelitian ini akurasi digunakan untuk menentukan kinerja dalam penelitian ini. Akurasi dapat dihitung menggunakan Persamaan 2.9 (Adinugroho & Sari, 2018).

$$Akurasi = \frac{TP+TN}{TP+TN+FN+FP} \quad (2.9)$$

Keterangan:
TP : nilai prediksi benar dan nilai sebenarnya benar
FP : nilai prediksi salah dan nilai sebenarnya salah
FN : nilai prediksi benar dan nilai sebenarnya salah
TN : nilai prediksi salah dan nilai sebenarnya benar

Pada penelitian ini *recall* digunakan untuk menghitung proporsi prediksi yang benar tanpa mempertimbangkan yang positif dan yang negatif, melainkan total keseluruhan.

BAB 3 METODOLOGI PENELITIAN

3.1 Studi Pustaka

Pengumpulan teori dan referensi dari penelitian sebelumnya yang menjadi dasar dalam membantu melakukan penelitian, atau kini lebih populer dengan sebutan studi pustaka. Dilihat dari fungsinya yakni untuk mempelajari penelitian terdahulu agar penelitian yang akan digunakan bisa memperbaiki kekurangan dari penelitian sebelumnya maka sudah seharusnya studi pustaka harus sesuai dengan permasalahan yang akan diangkat pada penelitian diantaranya:

1. *Preprocessing*
2. Segmentasi
3. *Erosion*
4. *Dilation*
5. *Blob Detection*
6. Deteksi Objek

3.2 Tipe Penelitian

Tipe penelitian pada penelitian ini menggunakan tipe non-implementatif dengan pendekatan analitik. Pendekatan analitik adalah pendekatan yang bertujuan untuk menjelaskan tentang tingkat ikatan dari antar bagian dalam topik penelitian dengan situasi tertentu yang sedang diteliti. Pendekatan ini akan menghasilkan sebuah produk hasil analisis.

3.3 Lokasi Penelitian

Laboratorium Komputasi Cerdas Fakultas Ilmu Komputer Universitas Brawijaya Malang adalah tempat yang akan digunakan untuk penelitian. Alasan memilih laboratorium tersebut karena untuk penentuan jumlah kendaraan menggunakan *Background Subtraction* dan *Blob Detection* memerlukan analisis untuk pemisahan objek dan *background* maka laboratorium komputasi cerdas sangat tepat untuk dijadikan sebagai lokasi penelitian.

3.4 Pengumpulan Data

Pada tahapan ini dilakukan pengumpulan data dari objek yang akan dilakukan penelitian. Data berupa *video* sebanyak 2 buah *video* yang diperoleh dari *Youtube* yang diunggah oleh Afrizal Firdaus tanggal 2 Januari 2016 dan diunduh pada tanggal 27 April 2018. Berikut ini daftar judul yang akan dilakukan penelitian.

- *Dataset* Jalan Tol Kopo 3 (3 jalur)
- *Dataset* Jalan Tol Kopo 1 (2 jalur)

Data tersebut merupakan *video* kendaraan yang melintas di jalan tol dengan kondisi jalan yang tidak terlalu padat kendaraan.

3.5 Perancangan Sistem

Perancangan sistem ini dilakukan ketika analisis kebutuhan sistem sudah ditentukan. Sistem pada penelitian ini dibangun dengan menggunakan Bahasa pemograman *Python*. Data yang digunakan dari penelitian ini adalah 3 buah *video* lalu lintas pada jalan tol yang bersumber dari *Youtube*. dari data tersebut dikonversi menjadi sekumpulan citra dan yang digunakan dalam penelitian sebanyak 10 citra yang diambil secara acak dan 1 buah citra *background* untuk setiap *video*. Proses pertama mendefinisikan dua buah citra sebagai *background* dan citra sebagai *testing.* Sebelum citra dilakukan penggabungan untuk melakukan pemisahan antara objek dan *background maka* dilakukan *preprocessing* yaitu mengkonversi citra berwarna menjadi citra keabuan. Setelah itu dilakukan pemisahan dengan mengurangi tiap piksel citra *testing* dengan tiap piksel dengan indek yang sama pada citra *background* yang dimutlakkan. Kemudian dilakukan segmentasi untuk mengelompokkan citra hasil pemisahan menjadi 2 grup yang dominan, yaitu objek dan *background* dengan memanfaatkan *threshold*. Setelah dilakukan segmentasi *Blob Detection* yaitu piksel yang nilainya sama dijadikan sebagai satu grup dan satu objek merepresentasikan satu *blob*.

Hasil keluaran sistem berupa jumlah yang merepresentasikan jumlah kendaran berdasarkan banyak *blob* yang terdeteksi.

3.6 Implementasi Sistem

Setelah proses perancangan selesai dilakukan, tahapan selanjutnya adalah melakukan implementasi sesuai dengan apa yang sudah dirancang sebelumnya. Pada sistem ini Implementasi yang akan diterapkan adalah dengan menggunakan metode Background Subtraction sebagai pemisah objek dengan background dan *Blob Detection* untuk menentukan piksel-piksel yang dianggap sebuah objek yang sama. Dalam implementasi sistem ini, spesifikasi kebutuhan perangkat dalam pembuatan sistem pada penelitian ini agar dapat berjalan dengan baik dibutuhkan suatu sistem yaitu sebagai berikut:

1. Spesifikasi Kebutuhan Perangkat Keras (*Hardware*) yaitu:
   a. *Processor* Intel(R) Core(TM) i5 CPU @2.70GHz (2 CPUs), ~2.7GHz.
   b. *Solid State Drive* 128 GB.
   c. Memori RAM 8192 MB.
2. Spesifikasi Kebutuhan Perangkat Lunak (*Software*) yaitu:
   a. macOS High Sierra 64-bit.
   b. Jupyter Notebook.
   c. Python 2.7.
   d. *Package* Python *OpenCV* untuk membaca *video* dan menyimpan *frame* tiap dalam satuan waktu.

3.7 Pengujian Sistem

Setelah tahap implementasi selesai maka tahap selanjutnya adalah pengujian sistem. Pada tahap pengujian sistem ini bertujuan untuk memastikan bahwa sistem sudah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang telah ditentukan. Pengujian dilakukan dengan menghitung hasil akurasi dari sistem dengan menggunakan *mean square error* dan *confusion matrix*.

3.8 Kesimpulan dan Saran

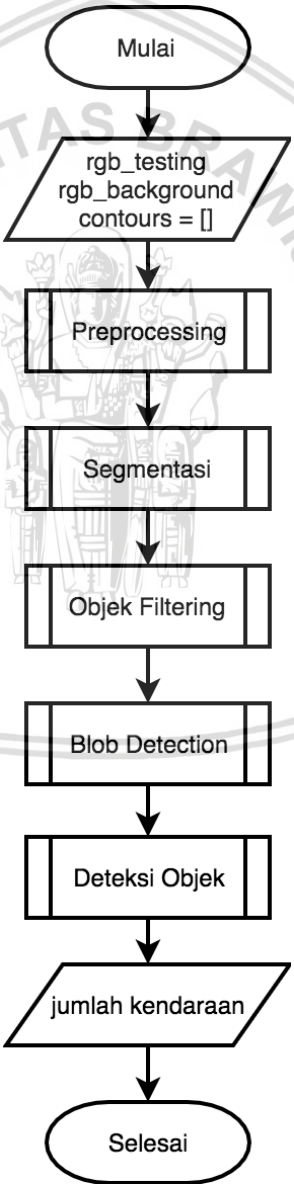
Pengambilan kesimpulan dilakukan setelah semua tahapan sebelumnya selesai dilakukan. Kesimpulan diambil berdasarkan dari analisis sistem terhadap perancangan yang dilakukan sebelumnya. Setelah kesimpulan selesai, dibutuhkan saran dengan harapan dapat memperbaiki kesalahan yang terjadi terhadap sistem.

BAB 4 PERANCANGAN

Pada bab ini berisi tentang tahap-tahap penelitian yang akan dilakukan. Tahap-tahap tersebut adalah Deskripsi Umum Sistem, Diagram Alir Sistem, Manualisasi, Perancangan Antarmuka dan Perancangan Pengujian.

4.1 Deskripsi Umum

Diagram alir digunakan untuk memudahkan proses dalam memahami alur sistem menggunakan suatu simbol-simbol yang mudah dipahami. Berikut beberapa langkah alur sistem pada penelitian ini, langkah awal yaitu masukkan citra berwarna yang dianggap sebagai citra *background* dan citra *testing*. Deskripsi umum dapat ditunjukkan Gambar 4.1.

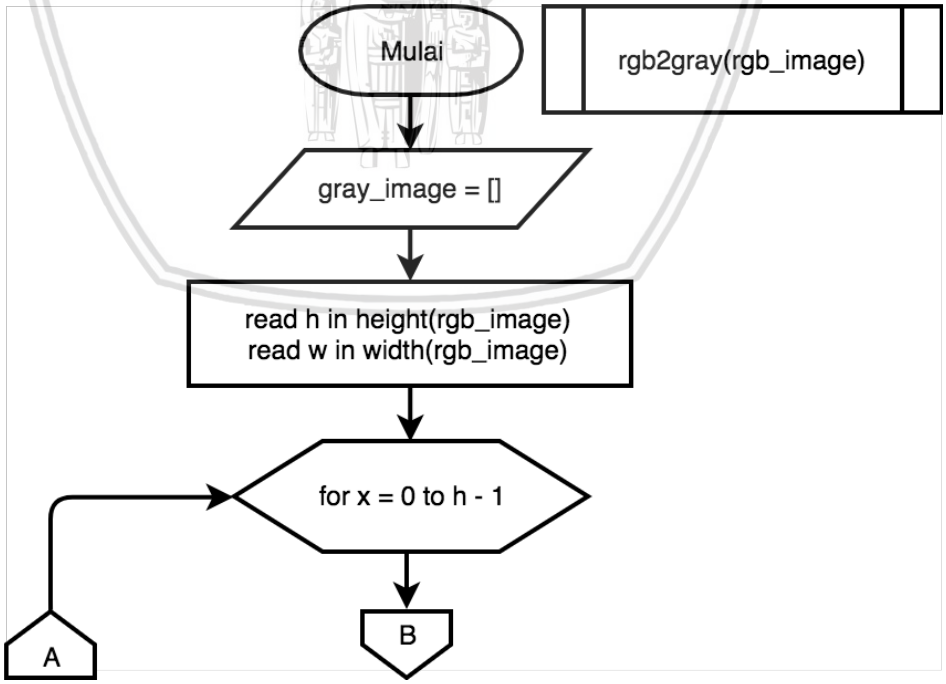


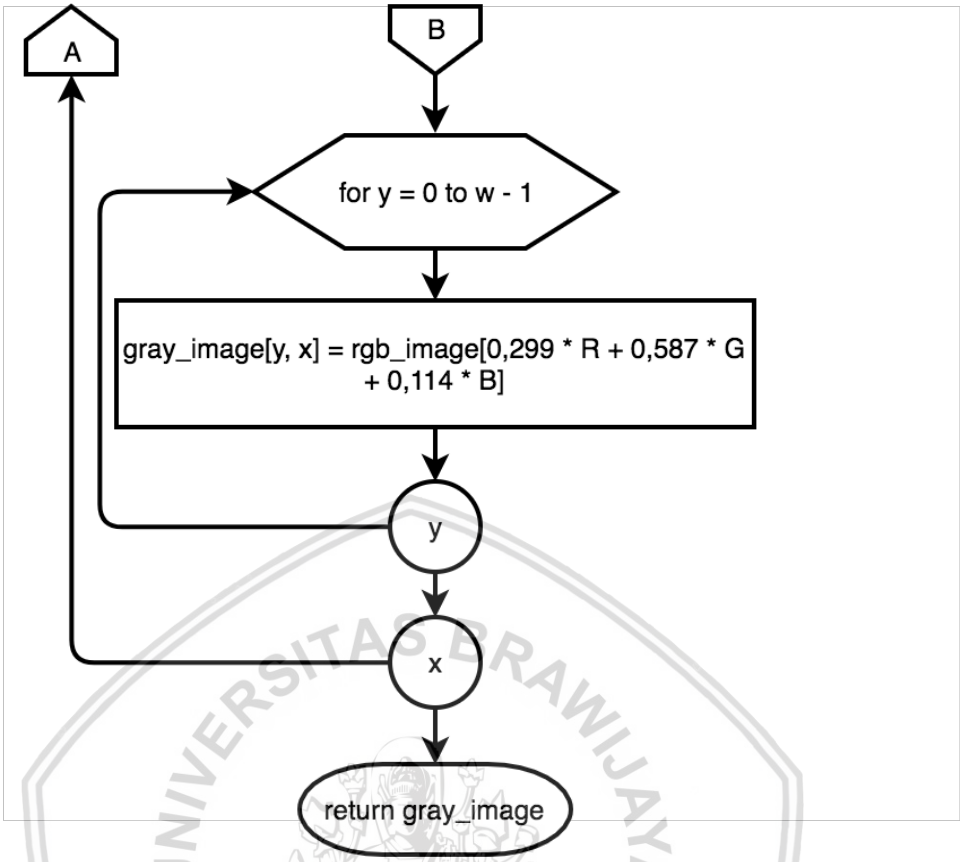
Gambar 4.1 *Flowchart* Deskripsi Umum

Pada Gambar 4.1, dilakukan input citra *background* dan citra *testing* dan variabel yang digunakan untuk menampung posisi dari tiap objek dalam suatu citra. Kedua citra tersebut dilakukan tahap *preprocessing* yaitu melakukan konversi dari citra berwarna menjadi citra keabuan (*grayscale*) atau citra 8-*bit*. Hasil dari *preprocessing* dilakukan *Background Subtraction* yaitu proses pemisahan objek dengan *background* dengan mengurangi tiap piksel objek dengan piksel *background* untuk setiap koordinat citra yang sama dan dilanjutkan proses segmentasi yaitu membagi citra menjadi 2 kelompok yang dominan. Proses selanjutnya dilakukan *object filtering* yaitu proses menonjolkan objek sehingga mudah dibedakan atau mengurangi derau dalam suatu citra. Proses terakhir adalah *Blob Detection* yaitu proses menentukan satu grup piksel yang berhubungan atau tidak, dengan menentukan piksel asal dan menelusuri piksel tetangga menggunakan relasi 8-*neighbors* dengan memberikan label jika piksel tersebut bernilai 255 dan menganggap piksel tersebut sebagai kesatuan piksel yang merepresentasikan suatu objek. Dari proses *Blob Detection* akan diketahui posisi objek dan jumlah objek yang sudah dilakukan penelusuran dan menentukan batas atas, bawah, kiri dan kanan untuk satu objek dalam suatu citra.

4.2 *Preprocessing*

Pada tahapan *preprocessing* digunakan untuk mempermudah pemisahan objek dengan *background*. Tahap ini hanya terdapat satu proses yaitu, mengubah citra berwana menjadi citra keabuan. Gambar dari alur tahapan *preprocessing* data input dapat ditunjukkan Gambar 4.2.





Gambar 4.2 *Flowchart Preprocessing*

Berikut adalah penjelasan mengenai proses konversi citra berwarna menjadi citra keabuan berdasarkan diagram alir pada Gambar 4.2:

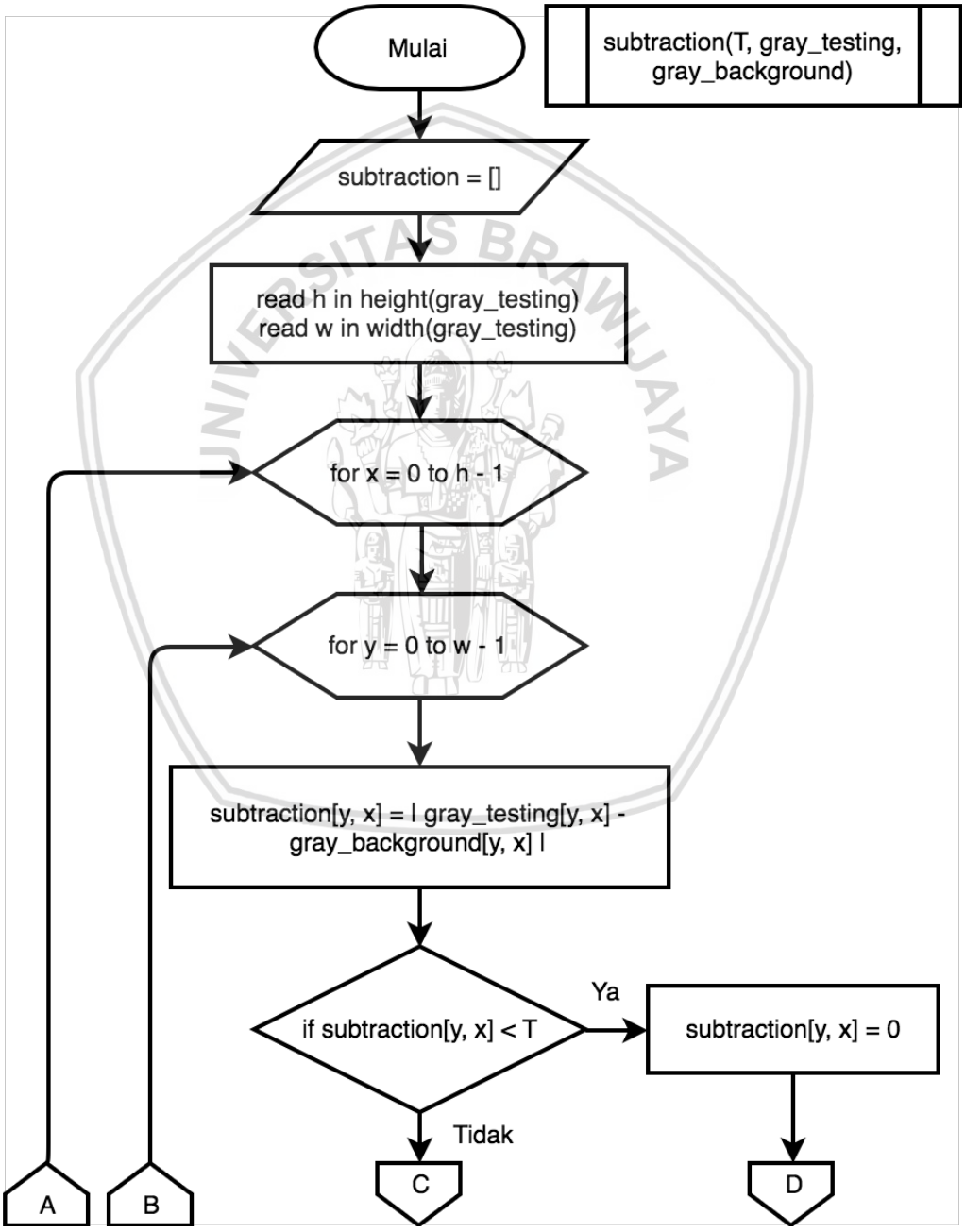
1. Inisialisasi variabel $gray\_image$ dengan matriks kosong
2. Proses membaca tinggi citra parameter variabel rgb_image ke dalam variabel h
3. Proses membaca lebar citra parameter variabel rgb_image ke dalam variabel w
4. Proses perulangan x untuk mengetahui posisi kolom matriks dari 0 hingga $h-1$
5. Proses perulangan y untuk mengetahui posisi baris matriks dari 0 hingga $w-1$
6. Proses perhitungan sekala keabuan dengan cara mengalikan nilai dari citra parameter variabel $rgb\_image(y,x)$, yang mana nilai indeks ke–0 dikalikan dengan 0,299, indeks ke–1 dikalikan dengan 0,587 dan indeks ke–2 dikalikan dengan 0,114. Hasil perkalian tiga indeks dilakukan penjumlahan dan disimpan ke dalam variabel $gray\_image(y,x)$
7. Keluaran dengan kembalian variabel $gray\_image$

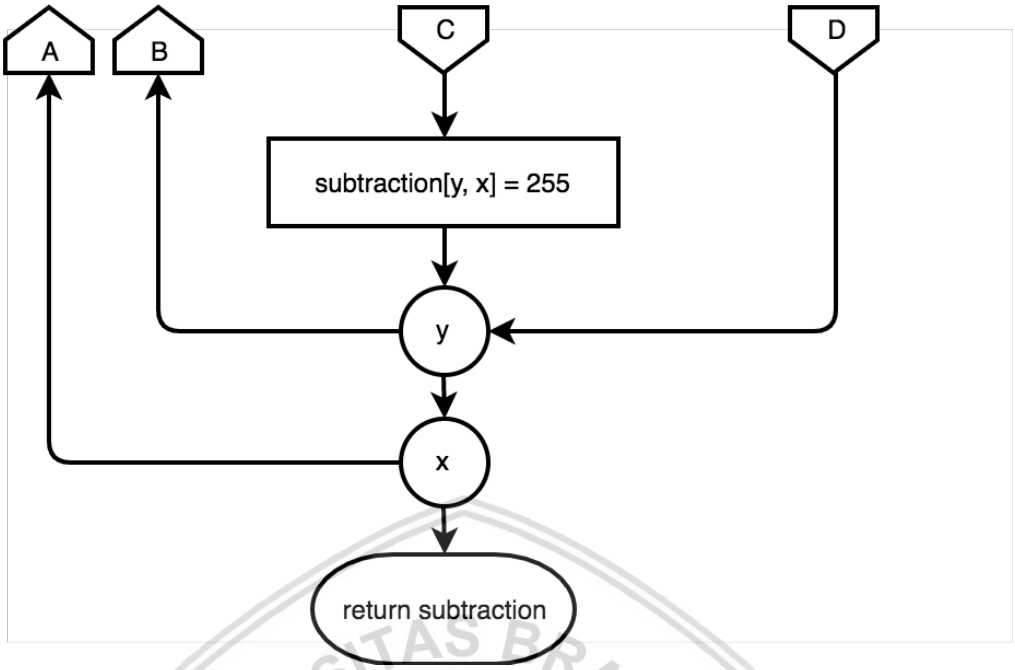
4.3 Segmentasi

Pada tahapan ini merupakan proses pemisahan objek dengan yang lainnya. Pada proses ini bertujuan untuk memisahkan objek dengan *background*.

4.3.1 *Background Subtraction*

Pada tahapan *Background Subtraction* dilakukan pemisahan antara objek dan *background* dengan mengurangi tiap piksel citra *testing* dengan citra *background* yang dimutlakkan. Kemudian menjadikan citra dibagi menjadi dua grup yang dominan dengan memanfaatkan batas batas ambang.





Gambar 4.3 *Flowchart Background Subtraction*

Berikut adalah penjelasan mengenai proses *Background Subtraction* berdasarkan diagram alir pada Gambar 4.3:

1. Inisialisasi variabel $subtraction$ dengan matriks kosong
2. Proses membaca tinggi citra parameter variabel $gray\_testing$ ke dalam variabel h
3. Proses membaca lebar citra parameter variabel $gray\_testing$ ke dalam variabel w
4. Proses perulangan x untuk mengetahui posisi kolom matriks dari 0 hingga $h-1$
5. Proses perulangan y untuk mengetahui posisi baris matriks dari 0 hingga $w-1$
6. Proses perhitungan *Background Subtraction* dengan cara mengurangkan nilai citra dari parameter variabel $gray\_testing(y, x)$ dengan nilai citra dari parameter variabel $gray\_background(y, x)$ yang dimutlakkan dan disimpan ke dalam variabel $subtraction(y, x)$
7. Proses kondisi apakah nilai citra dari parameter variabel $subtraction(y, x)$ kurang dari nilai parameter T
8. Jika Ya, simpan nilai 0 ke dalam variabel $subtraction(y, x)$
9. Jika Tidak, simpan nilai 255 ke dalam variabel $subtraction(y, x)$
10. Keluaran dengan kembalian variabel $subtraction$

4.4 *Object Filtering*

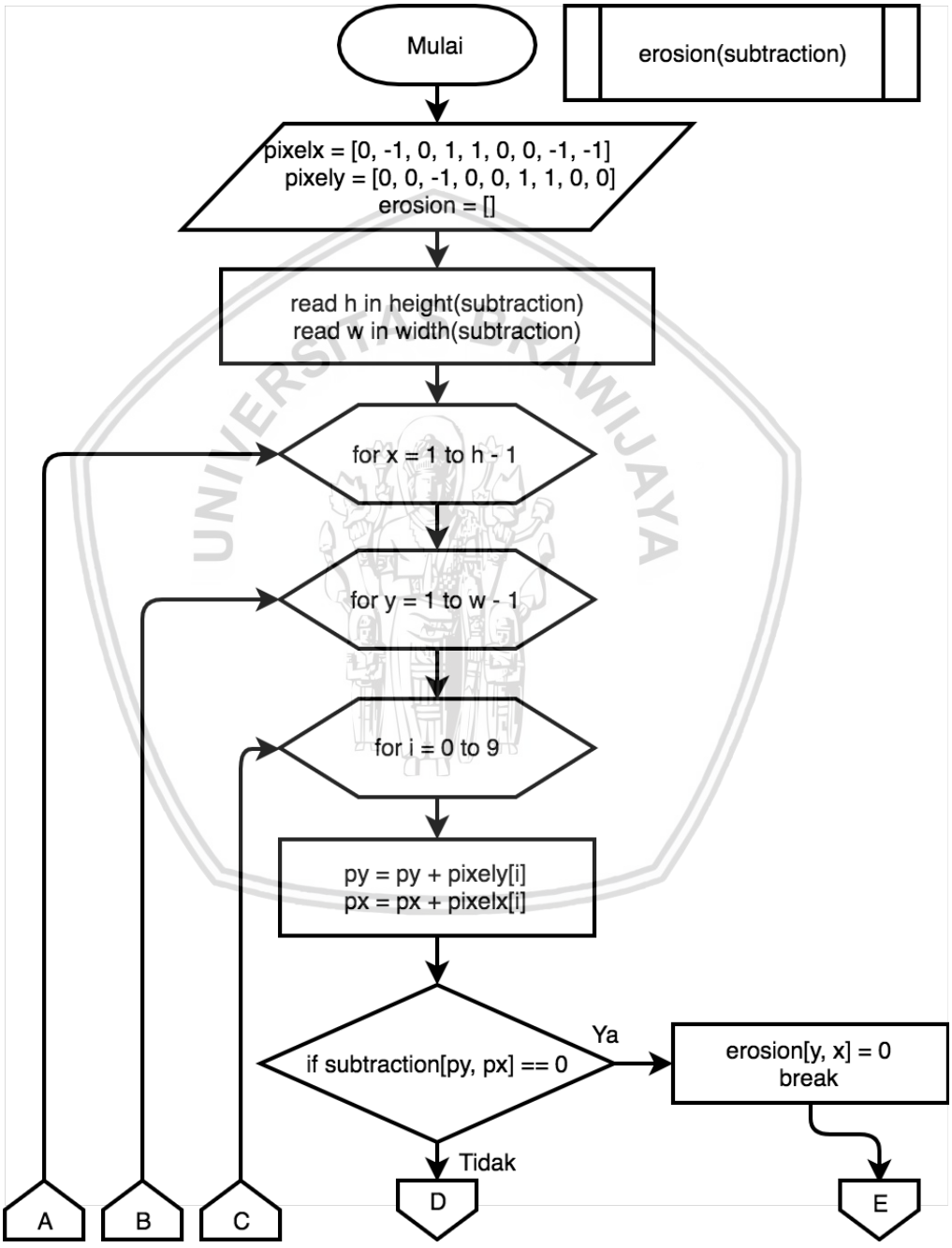
pada proses ini merupakan proses menonjolkan suatu objek pada citra sehingga lebih mudah dibedakan dengan objek lainnya. Pada proses ini bertujuan untuk mengurangi derau.

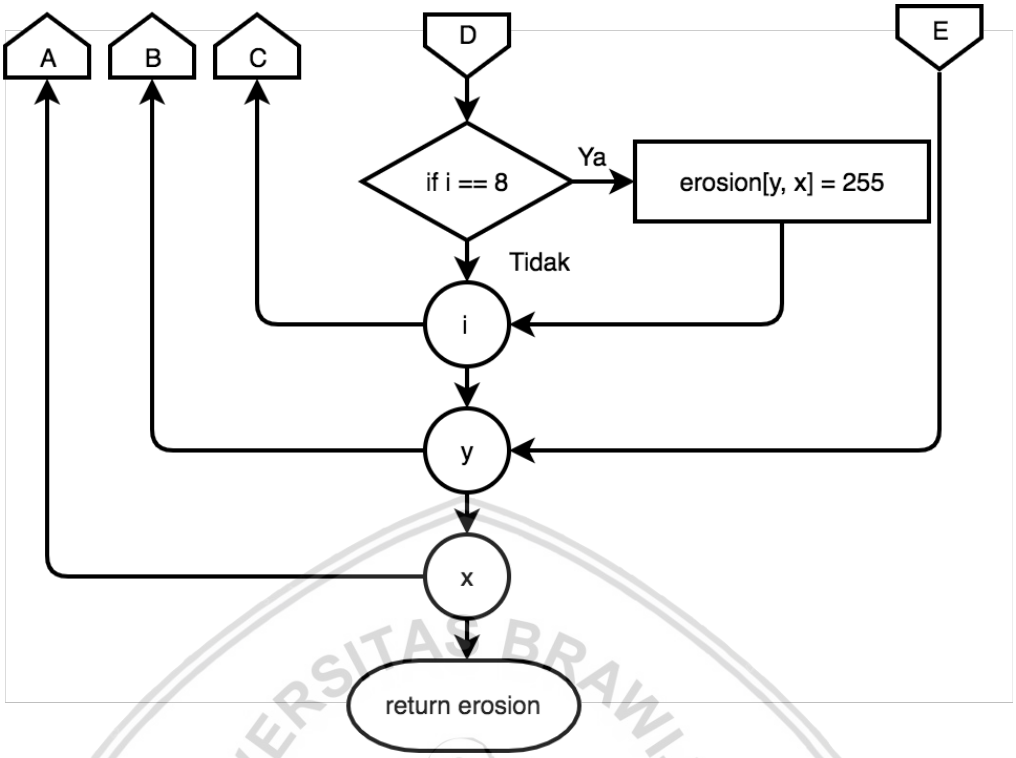
4.4.1 *Erosion*

Pada tahapan dilakukan proses menghilangkan derau yang bukan objek dengan memanfaatkan nilai piksel tetangganya.

1. *Erosion*

   Proses pemetaan terhadap piksel tetangganya dan mengganti nilai piksel dengan 0 jika nilai piksel tersebut atau piksel tetangganya bernilai 0.



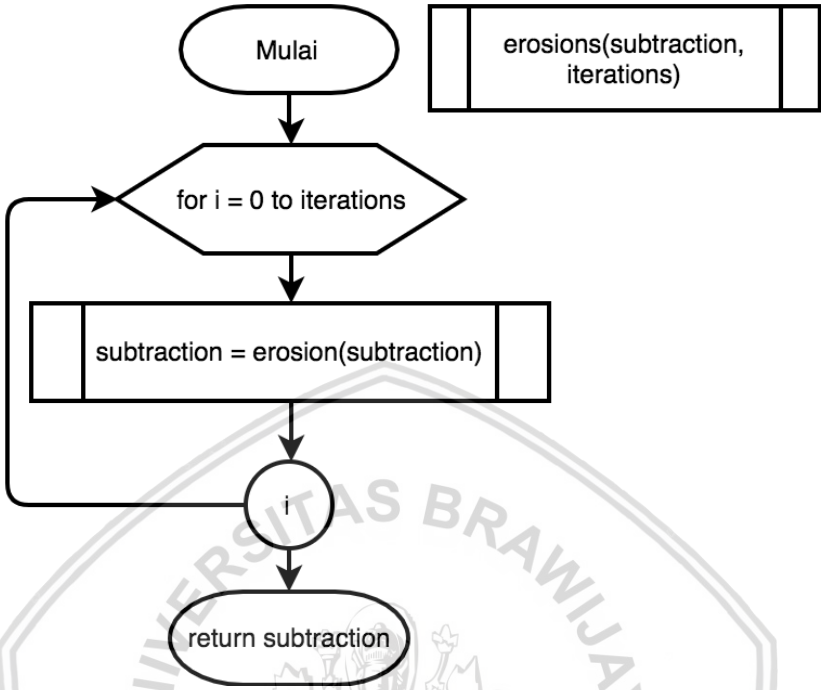


Gambar 4.4 *Flowchart Erosion*

Berikut adalah penjelasan mengenai proses *erosion* berdasarkan diagram alir pada Gambar 4.4:

1. Proses inisialisasi variabel $pixelx$ sama dengan matriks nilai 0, -1, 0, 1, 1, 0, 0, -1, -1 dan variabel $pixely$ sama dengan matriks nilai 0, 0, -1, 0, 0, 1, 1, 0, 0. Variabel $pixelx$ dan $pixely$ digunakan sebagai penentuan piksel tetangga dan inisialisasi variabel $erosion$ dengan matriks kosong
2. Proses membaca tinggi citra parameter variabel $subtraction$ ke dalam variabel h dan membaca lebar citra parameter variabel $subtraction$ ke dalam variabel w
3. Proses perulangan x untuk mengetahui posisi kolom matriks dari 0 hingga $h-1$
4. Proses perulangan y untuk mengetahui posisi baris matriks dari 0 hingga $w-1$
5. Proses perulangan i dari 0 hingga 9
6. Proses penjumlahan nilai parameter variabel y dengan nilai variabel $pixely(i)$ dan disimpan ke dalam variabel py dan penjumlahan nilai parameter variabel x dengan nilai variabel $pixelx(i)$ dan disimpan ke dalam variabel px
7. Proses kondisi apakah nilai citra dari parameter variabel $subtraction(y, x)$ sama dengan 0
8. Jika Ya, simpan nilai 0 ke dalam variabel $erosion(y, x)$ dan $break$
9. Jika Tidak dan proses kondisi i sama dengan 8, simpan nilai 255 ke dalam variabel $erosion(y, x)$
10. Keluaran dengan kembalian variabel $erosion$

2. *Erosions*
   Proses perulangan untuk melakukan proses sebanyak iterasi yang diinginkan.



Gambar 4.5 *Flowchart Erosions*

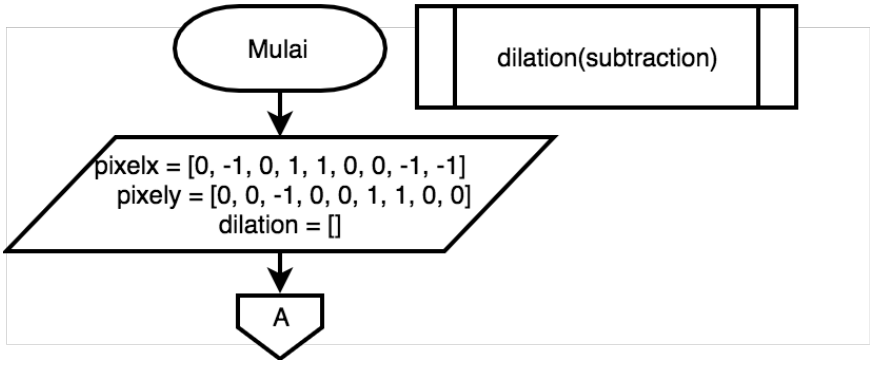
Berikut adalah penjelasan mengenai proses *erosions* berdasarkan diagram alir pada Gambar 4.5:

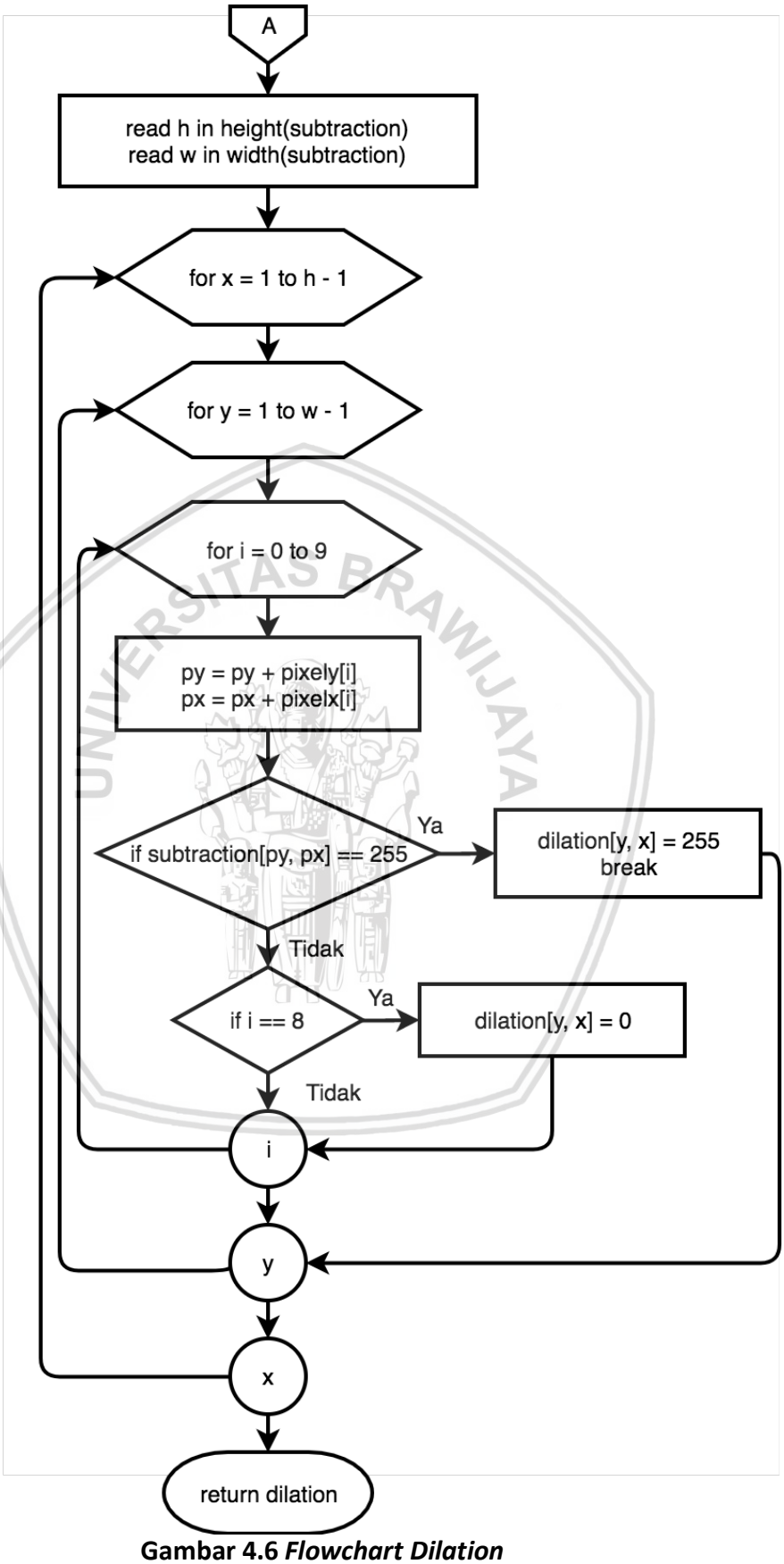
1. Proses perulangan i dari 0 hingga 9
2. Proses pemanggilan *method* $erosion$
3. Keluaran dengan kembalian variabel $subtraction$

4.4.2 *Dilation*

Pada tahapan dilakukan proses yang memungkinkan meperjelas objek dari hasil proses segmentasi dengan memanfaatkan nilai piksel tetangganya.

1. *Dilation*
   Proses pemetaan terhadap piksel tetangganya dan mengganti nilai piksel dengan 255 jika nilai piksel tersebut atau piksel tetangganya bernilai 255.





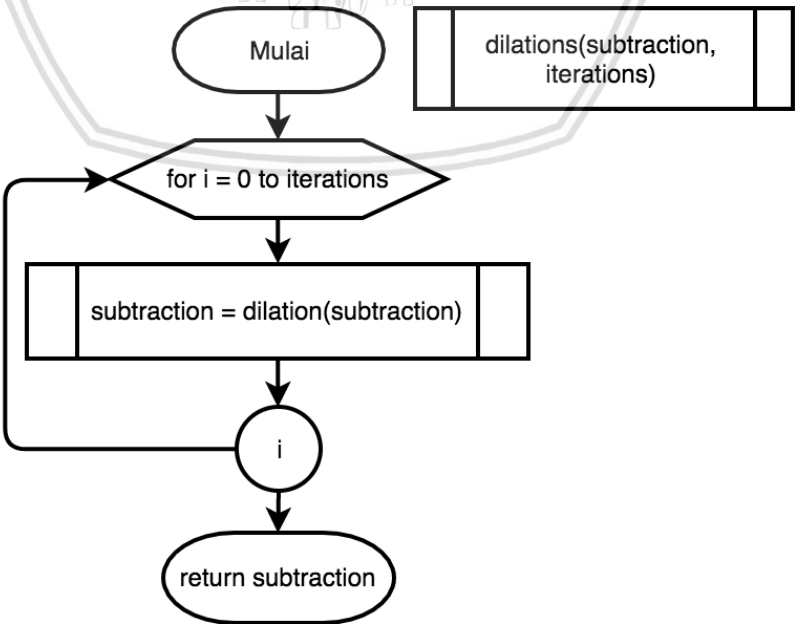
Gambar 4.6 *Flowchart Dilation*

Berikut adalah penjelasan mengenai proses *dilation* berdasarkan diagram alir pada Gambar 4.6:

1. Proses inisialisasi variabel $pixelx$ sama dengan matriks nilai 0, -1, 0, 1, 1, 0, 0, -1, -1 dan variabel $pixely$ sama dengan matriks nilai 0, 0, -1, 0, 0, 1, 1, 0, 0. Variabel $pixelx$ dan $pixely$ digunakan sebagai penentuan piksel tetangga dan inisialisasi variabel $dilation$ dengan matriks kosong
2. Proses membaca tinggi citra parameter variabel $subtraction$ ke dalam variabel h dan membaca lebar citra parameter variabel $subtraction$ ke dalam variabel w
3. Proses perulangan x untuk mengetahui posisi kolom matriks dari 0 hingga $h-1$
4. Proses perulangan y untuk mengetahui posisi baris matriks dari 0 hingga $w-1$
5. Proses perulangan i dari 0 hingga 9
6. Proses penjumlahan nilai parameter variabel y dengan nilai variabel $pixely(i)$ dan disimpan ke dalam variabel py dan penjumlahan nilai parameter variabel x dengan nilai variabel $pixelx(i)$ dan disimpan ke dalam variabel px
7. Proses kondisi apakah nilai citra dari parameter variabel $subtraction(y, x)$ sama dengan 255
8. Jika Ya, simpan nilai 0 ke dalam variabel $dilation(y, x)$ dan $break$, Jika Tidak dan proses kondisi i sama dengan 8, simpan nilai 0 ke dalam variabel $dilation(y, x)$
9. Keluaran dengan kembalian variabel $dilation$

2. Dilations

Proses pemetaan terhadap piksel tetangganya dan mengganti nilai piksel dengan 0 jika nilai piksel tersebut atau piksel tetangganya bernilai 0.



Gambar 4.7 *Flowchart Dilations*

Berikut adalah penjelasan mengenai proses *dilations* berdasarkan diagram alir pada Gambar 4.7:

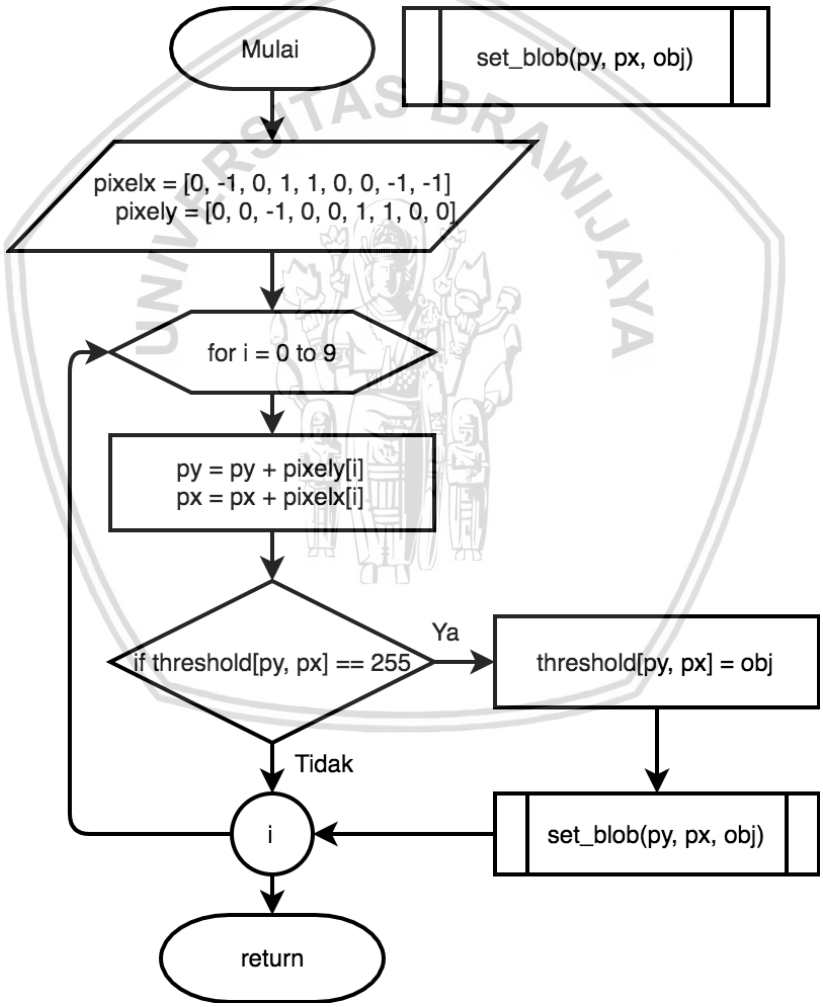
1. Proses perulangan i dari 0 hingga 9
2. Proses pemanggilan *method* $dilation$
3. Keluaran dengan kembalian variabel $subtraction$

4.5 *Blob Detection*

Pada tahapan ini merupakan proses pendeteksian objek serta memberikan label pada objek yang dianggap berbeda dengan menentukan piksel pertama dianggap objek dan memetakan piksel tetangganya yang dianggap satu objek.

1. *Set Blob*

   Proses pemetaan terhadap piksel tetangganya yang dianggap satu objek dan proses pelabelan jika piksel tersebut bernilai 255.



Gambar 4.8 *Flowchart Set Blob*

Berikut adalah penjelasan mengenai proses *Set Blob* berdasarkan diagram alir pada Gambar 4.5:

1. Proses inisialisasi variabel $pixelx$ sama dengan matriks nilai 0, -1, 0, 1, 1, 0, 0, -1, -1 dan variabel $pixely$ sama dengan matriks nilai 0, 0, -1, 0,

0, 1, 1, 0, 0. Variabel $pixelx$ dan $pixely$ digunakan sebagai penentuan piksel tetangga

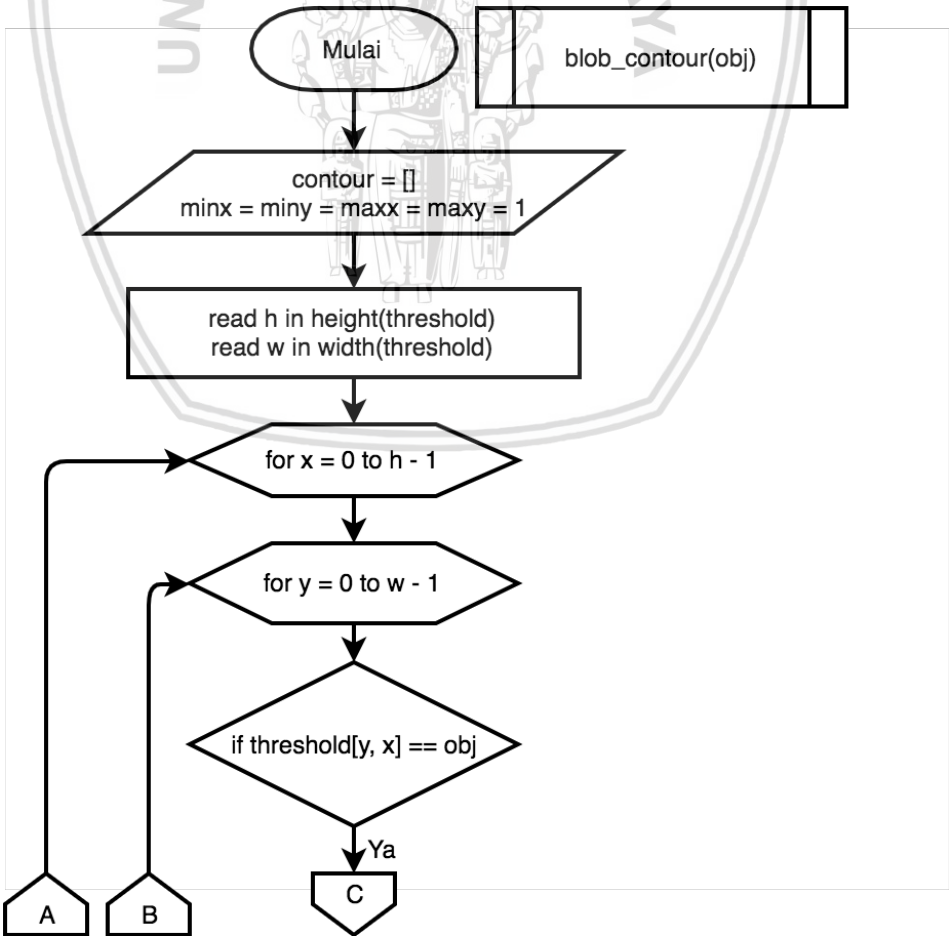
2. Proses perulangan i dari 0 hingga 9
3. Proses penjumlahan nilai parameter variabel py dengan nilai variabel $pixely(i)$ dan disimpan ke dalam variabel py dan penjumlahan nilai parameter variabel px dengan nilai variabel $pixelx(i)$ dan disimpan ke dalam variabel px
4. Proses kondisi apakah nilai citra dari parameter variabel $threshold(py, px)$ sama dengan 255
5. Jika Ya, simpan nilai parameter variabel obj ke dalam variabel $threshold(py, px)$ dan kemudian melakukan rekursif
6. Keluaran tanpa kembalian

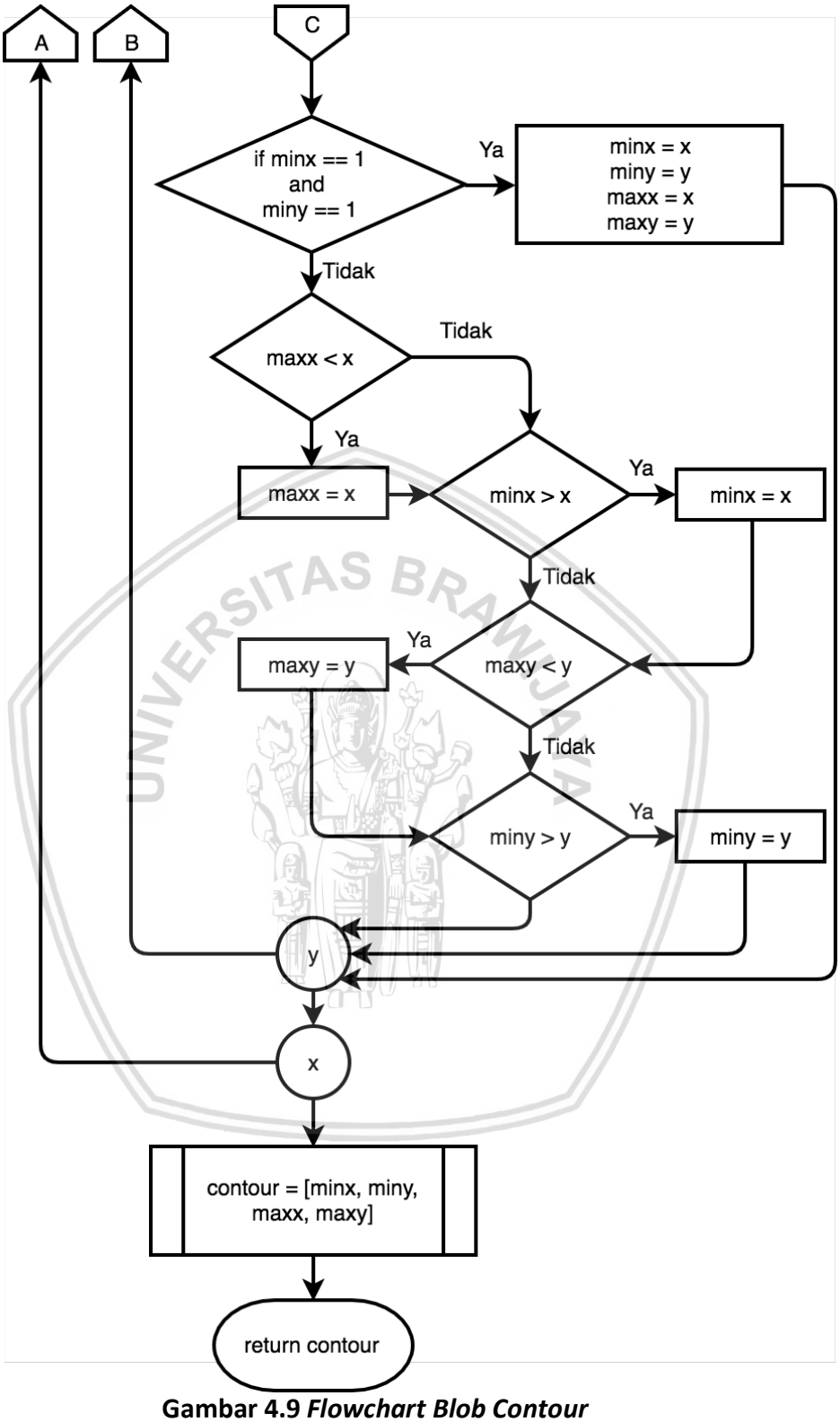
4.6 Deteksi Objek

Deteksi objek merupakan proses untuk mengetahui koordinat atau posisi objek sebelum dilakukan perhitungan jumlah objek pada suatu citra.

1. *Blob Contour*

Proses penentuan posisi suatu objek dengan mencari batas terkecil sumbu X, batas terbesar sumbu X, batas terkecil sumbu Y dan batas terbesar sumbu Y dan sekaligus mendapatkan jumlah objek yang sudah mengalami pelabelan.





Gambar 4.9 *Flowchart Blob Contour*

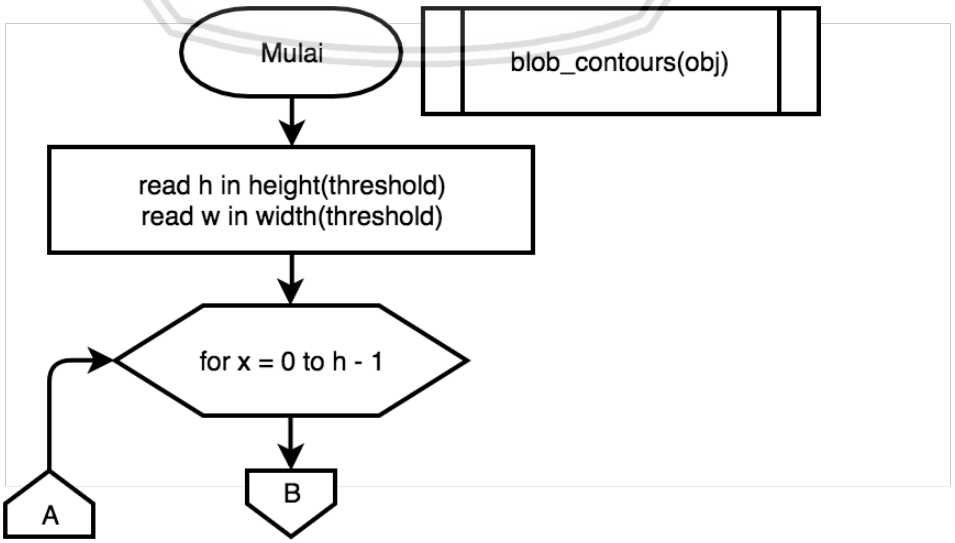
Berikut adalah penjelasan mengenai proses *Blob Contour* berdasarkan diagram alir pada Gambar 4.9:

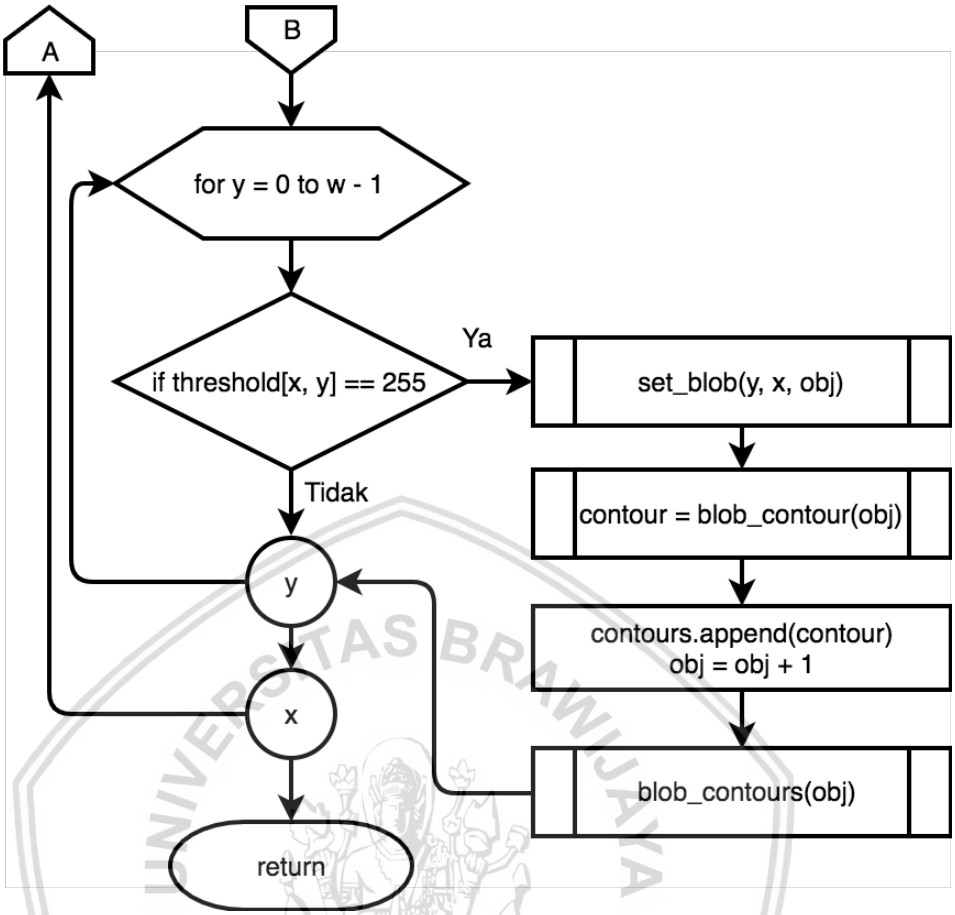
1. Proses inisialisasi variabel $contour$ sama dengan matriks kosong dan inisialisasi variabel $minx$, $maxx$, $miny$ dan $maxy$ sama dengan 1

2. Proses membaca tinggi citra parameter variabel $threshold$ ke dalam variabel h
3. Proses membaca lebar citra parameter variabel $threshold$ ke dalam variabel w
4. Proses perulangan x untuk mengetahui posisi kolom matriks dari 0 hingga $h-1$
5. Proses perulangan y untuk mengetahui posisi baris matriks dari 0 hingga $w-1$
6. Proses kondisi apakah nilai citra dari parameter variabel $threshold(x, y)$ sama dengan parameter variabel obj
7. Jika Ya, lakukan proses kondisi apakah variabel $minx$ dan $miny$ sama dengan 1
8. Jika kondisi nilai citra dari parameter variabel $threshold(x, y)$ sama dengan parameter variabel obj dan variabel $minx$ dan $miny$ sama dengan 1 terpenuhi, simpan nilai x ke dalam variabel $minx$, $maxx$ dan nilai y ke dalam variabel $miny$ dan $maxy$
9. Jika kondisi nilai citra dari parameter variabel $threshold(x, y)$ sama dengan parameter variabel obj terpenuhi dan variabel $minx$ dan $miny$ sama dengan 1 tidak terpenuhi, lakukan proses kondisi apakah $maxx$ kurang dari x, jika Ya simpan nilai x ke dalam $maxx$, jika $minx$ lebih dari x maka $minx$ sama dengan x, jika $maxy$ kurang dari y maka $maxy$ sama dengan y dan jika $miny$ lebih dari y maka $miny$ sama dengan y
10. Simpan variabel $minx$, $miny$, $maxx$ dan $maxy$ ke dalam matriks variabel $contour$
11. Keluaran dengan kembalian variabel $contour$

2. *Blob Contours*

Proses penentuan titik awal sebelum dilakukan pemetaan terhadap piksel tetangganya dari titik awal yang sudah ditentukan dan pelabelan terhadap objek yang berbeda.





Gambar 4.10 *Flowchart Blob Contours*

Berikut adalah penjelasan mengenai proses *Blob Contours* berdasarkan diagram alir pada Gambar 4.10:

1. Proses membaca tinggi citra parameter variabel $threshold$ ke dalam variabel h
2. Proses membaca lebar citra parameter variabel $threshold$ ke dalam variabel w
3. Proses perulangan x untuk mengetahui posisi kolom matriks dari 0 hingga $h-1$
4. Proses perulangan y untuk mengetahui posisi baris matriks dari 0 hingga $w-1$
5. Proses kondisi apakah nilai citra dari parameter variabel $threshold(x, y)$ sama dengan 255
6. Jika Ya, memanggil fungsi set_blob untuk melakukan proses pada Gambar 4.5. Kemudian memanggil fungsi $blob\_contour$ untuk melakukan proses pada Gambar 4.6. Kemudian melakukan penambahan indeks matriks dari proses fungsi $blob\_contour$, variabel obj ditambah dengan 1 dan lakukan proses rekursif
7. Keluaran dengan tanpa kembalian

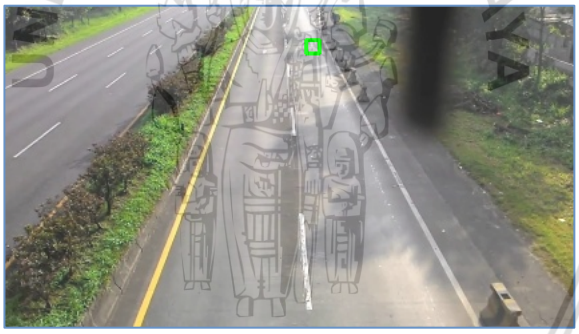
4.7 Manualisasi

Perhitungan ini dilakukan sebagai gambaran umum dalam pengerjaan penelitian ini. Ada beberapa tahapan yang akan dilakukan perhitungan manual yaitu, manualisasi *preprocessing*, segmentasi, *object* filtering, *Blob Detection* dan deteksi objek.

4.7.1 Penentuan Citra

Pada tahap ini dilakukan penentuan citra berwarna yang digunakan sebagai *background* dan citra *testing* untuk dilakukan pengujian. Citra *background* diambil dari tiap *dataset* yang tidak terdapat kendaraan dan citra *testing* diambil dari citra acak dari sebuah *video dataset* penelitian.

Gambar 4.11 Citra *Testing*



Gambar 4.12 Citra *Background*

Pada Gambar 4.11 merupakan contoh citra *testing* dan Gambar 4.12 contoh citra yang dianggap sebagai citra *background*. Pada Gambar 4.13 merupakan potongan dari citra *testing* bagian warna hijau dengan ukuran piksel 10*x*10 dan citra *background* pada Gambar 4.14 dengan posisi piksel yang sama untuk dilakukan proses manualisasi.

Gambar 4.14
Manualisasi *Testing*

Gambar 4.13
Manualisasi *Background*

Sebelum dilakukan *preprocessing* dilakukan pengambilan nilai pada citra *testing* ataupun citra *background* di setiap pikselnya. Pada citra berwarna mempunyai 3 nilai yaitu R, G dan B yang dapat dilihat pada Tabel 4.2 sebagai nilai dari citra *testing* dan Tabel 4.3 nilai dari citra *background*. Keterangan tiap piksel dapat dilihat pada Tabel 4.1.

Tabel 4.1 Struktur piksel R, G dan B

R G B

Tabel 4.2 Data Citra *Testing*

183 161 150	187 165 154	184 165 152	189 169 158	208 189 181	183 165 158	222 208 202	238 226 224	186 178 178	190 184 185
255 243 232	255 237 226	255 237 226	253 232 224	248 231 222	221 205 198	228 216 212	236 224 224	189 181 182	195 188 191
175 155 144	184 164 153	180 160 149	175 157 146	166 149 140	157 141 134	180 168 164	236 224 224	197 189 190	201 194 197
107 85 73	102 83 70	102 83 70	92 75 62	94 78 66	106 90 83	193 181 177	230 220 220	191 182 185	201 194 199
142 120 108	128 106 94	135 116 103	151 132 119	159 143 131	223 207 200	180 168 164	118 108 108	201 194 197	203 196 201
189 164 154	189 164 154	157 135 124	108 88 77	96 77 69	85 69 62	62 50 46	68 58 58	192 186 187	207 200 203
76 53 45	81 59 48	71 51 40	65 46 38	80 63 54	76 62 56	96 85 81	62 52 52	181 175 176	202 195 198
72 56 44	75 58 49	70 55 46	78 62 55	79 65 59	81 70 66	117 108 105	77 69 69	175 170 169	201 195 196
92 80 70	92 80 70	92 81 73	77 65 59	73 62 58	74 65 62	71 64 61	51 43 43	177 172 171	199 194 195
87 76 68	88 77 69	80 68 62	80 70 63	80 71 67	68 59 56	75 67 67	53 47 48	185 180 181	226 221 222

Tabel 4.3 Data Citra *Background*

205 200 201	209 204 205	209 204 205	209 204 205	211 206 207	213 208 209	214 209 210	215 210 211	198 193 194	184 179 180
211 209 209	213 208 209	211 206 207	214 209 210	220 215 216	224 219 220	226 221 222	221 216 217	209 204 205	197 192 193
201 199 199	203 198 199	206 201 202	214 209 210	221 216 217	223 218 219	223 218 219	218 213 214	213 208 209	200 195 196
186 186 186	197 195 195	208 206 206	215 213 213	214 212 212	209 207 207	207 205 205	208 206 206	203 201 201	198 196 196
210 210 210	220 218 218	230 228 228	233 231 231	228 226 226	225 223 223	227 225 225	222 220 220	214 212 212	210 208 208
226 226 226	224 224 224	222 222 222	219 218 220	213 212 214	213 210 212	215 212 214	218 215 217	214 211 213	209 207 207
194 194 194	196 194 194	198 198 198	204 203 205	206 205 207	208 205 207	209 206 208	207 204 206	210 207 209	209 207 207
186 184 183	190 186 185	192 190 189	196 194 194	201 199 199	204 201 203	205 202 204	208 203 205	210 205 207	210 205 207
185 181 180	187 183 182	189 185 184	191 189 189	196 194 194	199 196 198	200 197 199	203 198 200	206 201 203	211 206 208
183 179 178	186 182 181	189 185 184	195 190 191	202 197 198	208 203 205	212 207 209	220 215 217	225 220 222	231 226 228

4.7.2 *Preprocessing*

Proses pada *preprocessing* adalah proses yang menghasilkan citra keabuan pada citra *testing* ataupun citra *background*. Setiap *cell* pada Tabel 4.2 citra *testing* dan Tabel 4.3 citra *background* akan dilakukan *preprocessing* sesuai dengan Persamaan 2.1. *Preprocessing* dilakukan nilai matriks pada Tabel 4.2 kolom 1 baris 1.

Preprocessing = (0,299 * 183) + (0,587 * 161) + (0,114 * 150)
= 54,717 + 94,507 + 17,1
= *round*(166,324)
= 166

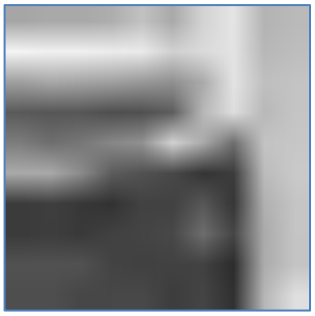
Demikian seterusnya hingga kolom dan baris terakhir dari matriks, sehingga diperoleh citra keabuan dan dapat dilihat pada Tabel 4.4 untuk citra *testing* dan Tabel 4.5 untuk citra *background*.

Tabel 4.4 Hasil Perhitungan *Preprocessing* Citra *Testing*

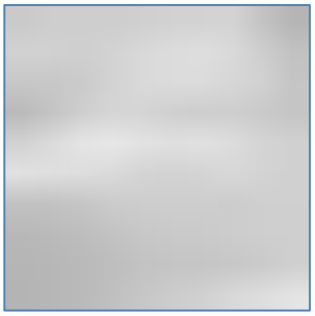
166	170	169	174	194	170	212	229	180	186
245	241	241	237	235	209	219	228	184	190
160	169	165	161	153	145	171	228	192	196
90	87	87	79	81	94	184	223	185	197
125	111	120	136	146	211	171	111	196	199
170	170	140	93	82	73	53	61	188	202
59	64	56	51	67	66	88	55	177	197
59	62	58	66	69	73	110	71	171	197
82	82	83	68	65	67	66	45	173	196
78	79	71	72	73	61	69	49	182	223

Tabel 4.5 Hasil Perhitungan *Preprocessing* Citra *Background*

202	206	206	206	208	210	211	212	195	181
210	210	208	211	217	221	223	218	206	194
200	200	203	211	218	220	220	215	210	197
186	196	207	214	213	208	206	207	202	197
210	219	229	232	227	224	226	221	213	209
226	224	222	219	213	211	213	216	212	208
194	195	198	204	206	206	207	205	208	208
184	187	190	195	200	202	203	205	207	207
182	184	186	190	195	197	198	200	203	208
180	183	186	192	199	205	209	217	222	228



Gambar 4.15
Preprocessing Testing



Gambar 4.16
Preprocessing Background

Pada hasil *preprocessing* hanya mempunyai 1 nilai skala 0 sampai 255 yang dapat dilihat pada Gambar 4.15 sebagai hasil *preprocessing* dari citra *testing* dan Gambar 4.16 hasil *preprocessing* dari citra *background*.

4.7.3 Segmentasi

Proses pada segmentasi adalah proses pemisahan objek dengan *background* pada citra hasil *preprocessing* dengan mengurangi tiap *cell* yang sama pada citra *testing* dan citra *background* dan dimutlakkan agar tidak menghasilkan *cell* bernilai negative dan hasil pemisahan dibagi menjadi dua kelompok yang dominan.

Pada Tabel 4.4 nilai tiap *cell* yang sama dikurangi dengan nilai tiap *cell* Tabel 4.5, hasil pengurangan dimutlakkan dan melewati proses kondisi jika hasil mutlak lebih besar dari *threshold* nilai *cell* sama dengan 255, jika kurang dari *threshold* nilai *cell* sama dengan 0 menggunakan Persamaan 2.2. *Background Subtraction* nilai matriks pada kolom 1 baris 1 dengan menggunakan *threshold* sama dengan 30 berdasarkan observasi yang memperoleh akurasi yang paling baik.

$$\textit{Subtraction} = |\ 166 - 202\ | > 30$$
$$= |\ -36\ | > 30$$
$$= 255$$

Demikian seterusnya hingga kolom dan baris terakhir dari matriks, sehingga diperoleh sebuah citra hasil dari proses *Background Subtraction* dapat dilihat pada Tabel 4.6.

Tabel 4.6 Hasil Perhitungan *Background Subtraction*

255	255	255	255	0	255	0	0	0	0
255	255	255	0	0	0	0	0	0	0
255	255	255	255	255	255	255	0	0	0
255	255	255	255	255	255	0	0	0	0
255	255	255	255	255	0	255	255	0	0
255	255	255	255	255	255	255	255	0	0
255	255	255	255	255	255	255	255	255	0
255	255	255	255	255	255	255	255	255	0
255	255	255	255	255	255	255	255	0	0
255	255	255	255	255	255	255	255	255	0

Gambar 4.17 *Background Subtraction*

Pada hasil *Background Subtraction* hanya mempunyai nilai piksel 0 atau 255, piksel bernilai 0 menunjukkan piksel bagian dari *background*, sedangkan piksel bernilai 255 bagian dari objek yang dapat dilihat pada Gambar 4.17.

4.7.4 *Object Filtering*

Proses pada *object filtering* adalah proses mengubah nilai suatu *cell* pada matriks dan *kernel* yang dipakai proses ini adalah *kernel* 3x3 dengan iterasi sebanyak 2 untuk proses *erosion* dan iterasi sebanyak 4 untuk proses *dilation*.

1. *Erosion*

   Erosion merupakan proses mengubah nilai berdasarkan nilai terkecil hasil perkalian citra dengan *kernel morfologi*.

   Pada Tabel 4.6 dilakukan perkalian dengan *kernel* morfologi 3x3 pada kolom 2 bari 2.

$$\begin{aligned} \textit{Erosion} &= \textit{min}(255 \times 1,\ 255 \times 1,\ 255 \times 1, \\ &\qquad 255 \times 1,\ 255 \times 1,\ 255 \times 1, \\ &\qquad 255 \times 1,\ 255 \times 1,\ 255 \times 1) \\ &= 255 \end{aligned}$$

   Demikian seterusnya hingga kolom dan baris terakhir dari matriks dan diulangi lagi sebanyak iterasi, sehingga diperoleh sebuah citra hasil dari proses *erosion* dapat dilihat pada Tabel 4.7.

Tabel 4.7 Hasil Perhitungan *Erosion*

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	255	0	0	0	0	0	0	0
0	0	255	0	0	0	0	0	0	0
0	0	255	255	0	0	0	0	0	0
0	0	255	255	255	0	255	0	0	0
0	0	255	255	255	255	255	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 4.18 *Erosion*

Pada hasil *erosion* terjadi perubahan nilai piksel dari awalnya 255 menjadi 0 yang dapat dilihat pada Gambar 4.18.

2. *Dilation*

Dilation merupakan proses mengubah nilai berdasarkan nilai terbesar hasil perkalian citra dengan *kernel morfologi*.

Pada tabel 4.7 dilakukan perkalian dengan *kernel morfologi* 3x3 pada kolom 1 baris 1.

Erosion = *max*(0 x 1, 0 x 1, 0 x 1,
0 x 1, 0 x 1, 0 x 1,
0 x 1, 0 x 1, 0 x 1)
= 0

Demikian seterusnya hingga kolom dan baris terakhir dari matriks dan diulangi lagi sebanyak iterasi, sehingga diperoleh sebuah citra hasil dari proses *erosion* dapat dilihat pada Tabel 4.8.

Tabel 4.8 Hasil Perhitungan *Dilation*

0	0	0	0	0	0	0	0	0	0
0	255	255	255	255	0	0	0	0	0
0	255	255	255	255	255	255	0	0	0
0	255	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	0	0	0	0	0	0	0	0	0

Gambar 4.19 *Dilation*

Pada hasil *dilation* terjadi perubahan nilai piksel dari awalnya 0 menjadi 255 yang dapat dilihat pada Gambar 4.19.

4.7.5 *Blob Detection*

Proses *Blob Detection* adalah proses mengubah *cell* yang berniai 255 dan *cell* tetangganya dengan memberikan label yang sama pada *cell* tersebut yang merepresentasikan sekumpulan *cell* dalam sebuah matriks.

Pada Tabel 4.8 nilai tiap *cell* baris 1, baris 10, kolom 1 dan kolom 10 dilakukan penggantian nilai dengan nilai 0 untuk mengatasi kolom tidak terdefinisi ketika melakukan proses *Blob Detection* dapat dilihat pada Tabel 4.9.

Tabel 4.9 Sampel Citra *Blob Detection*

0	0	0	0	0	0	0	0	0	0
0	255	255	255	255	0	0	0	0	0
0	255	255	255	255	255	255	0	0	0
0	255	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	0	0	0	0	0	0	0	0	0

1. Pencarian

   Pada Tabel 4.9 dilakukan proses pencarian *cell* asal bernilai 255 yang digunakan sebagai acuan untuk mendapatkan posisi dari sebuah objek. Proses penentuan *cell* asal pada matriks seperti pada Tabel 4.10.

Tabel 4.10 Pencarian

0	0	0	0	0	0	0	0	0	0
0	255	255	255	255	0	0	0	0	0
0	255	255	255	255	255	255	0	0	0
0	255	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0

0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	0	0	0	0	0	0	0	0	0

2. Pelabelan

Pada Tabel 4.10 *cell* berwarna merah merupakan *cell* asal setelah dilakukan proses pencarian *cell* asal bernilai 255. Proses pelabelan ini dilakukan dengan mengubah nilai *cell* dengan sebuah nilai unik selain 0 atau 255 seperti pada Tabel 4.11.

Tabel 4.11 Pelabelan

0	0	0	0	0	0	0	0	0	0
0	1	255	255	255	0	0	0	0	0
0	255	255	255	255	255	255	0	0	0
0	255	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	0	0	0	0	0	0	0	0	0

3. *Clockwise*

Proses *Clockwise* merupakan proses pengecekan *cell* tetangga yang bernilai 255 searah jarum jam pada *cell* asal yang sudah dilakukan pelabelan. Pada Tabel 4.11 dilakukan *clockwise* pada kolom 2 baris 2 menggunakan relasi 8-*neigbour* pada Tabel 2.1.

Proses *Clockwise* dapat dilihat pada Tabel 4.12 yang mana warna biru merupakan *cell* yang dilakukan pengecekan dan warna kuning merupakan *cell* yang bernilai 255 untuk dilakukan pelabelan dan pengulangan proses *clockwise* dengan *cell* pada warna kuning tersebut.

Tabel 4.12 *Clockwise*

0	0	0	0	0	0	0	0	0	0
0	255	255	255	255	0	0	0	0	0
0	255	255	255	255	255	255	0	0	0
0	255	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	0	0	0	0	0	0	0	0	0

Demikian seterusnya hingga proses *clockwise* pada tiap *cell* selesai melakukan pengecekan ke semua *cell* tetangganya. Hasil dari proses *Blob Detection* dapat dilihat pada Tabel 4.13.

Tabel 4.13 *Blob Detection*

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0

4.7.6 Deteksi Objek

Proses deteksi objek adalah proses menentukan posisi sekumpulan *cell* yang sudah dilakukan pelabelan pada Tabel 4.13 dengan menentukan koordinat sumbu X terkecil, koordinat sumbu Y terkecil, koordinat sumbu X terbesar dan koordinat sumbu Y terbesar dari sekumpulan *cell* tersebut.

1. Pencarian

   Pada Tabel 4.13 dilakukan proses pencarian *cell* asal bernilai 1 yang digunakan sebagai acuan untuk mendapatkan posisi dari sebuah objek. Proses penentuan *cell* asal pada matriks seperti pada Tabel 4.14.

Tabel 4.14 Pencarian

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0

2. Inisialisasi Posisi Awal

   Pada Tabel 4.14 didapatkan posisi awal pada kolom 2 baris 2 pada koordinat (1, 1), maka:

   $minx$ = 1

   $miny$ = 1

   $maxx$ = 1

$maxy$ = 1

Keterangan:
$minx$: koordinat terkecil piksel pada sumbu X
$miny$: koordinat terkecil piksel pada sumbu Y
$maxx$: koordinat terbesar piksel pada sumbu X
$maxy$: koordinat terbesar piksel pada sumbu Y

3. *Update* Posisi

Update posisi terjadi jika koordinat sumbu X terkecil lebih besar dari koordinat sumbu X saat ini, jika koordinat sumbu X terbesar kurang dari koordinat sumbu X saat ini, jika koordinat sumbu Y terkecil lebih besar dari koordinat sumbu Y saat ini dan jika koordinat sumbu Y terbesar kurang dari koordinat sumbu Y saat ini.

Demikian seterusnya hingga kolom dan baris terakhir dari matriks, sehingga diperoleh koordinat terkecil yaitu koordinat(1, 1) dan koordinat terbesar yaitu koordinat(8, 8) seperti pada Tabel 4.15.

Tabel 4.15 Deteksi Objek

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0

Gambar 4.20 *Blob Detection*

Hasil dari sistem berupa citra yang sudah ditandai dengan warna hijau untuk objek yang terdeteksi dari hasil pencarian koordinat X dan koordinat Y terkecil dan koordinat X dan koordinat Y terbesar dari setiap objek dan jumlah objek disuatu citra tertentu seperti pada Gambar 4.20.

4.8 Implementasi

Proses implementasi pada bab ini adalah tahap pembuatan sistem "Penentuan Jumlah Kendaraan Menggunakan *Blob Detection* dan *Background Subtraction*" berdasarkan perancangan yang telah dibuat sebelumnya. Dalam bab ini akan berisi mengenai terkait kode program dari sistem yang mengacu pada perancangan proses pada bab sebelumnya. Dimana meliputi *preprocessing,* segmentasi, *object filtering*, *Blob Detection* dan deteksi objek.

4.8.1 Implementasi Proses *Preprocessing*

```
1   def rgb2gray(rgb):
2       return np.dot(rgb[...,:3], [0.299, 0.587, 0.114])
```

Kode Program 4.1 Proses *Preprocessing*

Penjelasan Kode Program 4.1, pada baris ke:

1. *Method* `rgb2gray` dengan parameter `rgb` sebagai matriks 2 dimensi dari citra berwarna.
2. Fungsi `dot` dari *numpy* untuk memproses setiap *cell* dari matriks, yang mana mengalikan *cell* indeks ke 1 dengan 0.299, indeks ke 2 dengan 0,587 dan indeks ke 3 dengan 0,114.

4.8.2 Implementasi Proses Segmentasi

```
1   def subtraction(T, gray_img, gray_background):
2       h = gray_img.shape[0]
3       w = gray_img.shape[1]
4       subtraction = np.asarray(np.zeros_like(gray_img),
    dtype="uint8")
5
6       for y in range(0, h):
7           for x in range(0, w):
8               subtraction[y, x] = abs(gray_img[y, x] -
    gray_background[y, x])
9               subtraction[y, x] = 255 if subtraction[y, x]
    >= T else 0
10
11      return subtraction
```

Kode Program 4.2 Proses *Background Subtraction*

Penjelasan Kode Program 4.2, pada baris ke:

1. *Method* `subtraction` dengan parameter `T` sebagai *threshold,* matriks 2 dimensi dari `gray_img` sebagai citra keabuan *testing* dan `gray_background` sebagai citra keabuan *background*.
2. Inisialisasi variabel `h` bernilai tinggi dari matriks `gray_img`.
3. Inisialisasi variabel `w` bernilai lebar dari matriks `gray_img`.
4. Inisialisasi variabel `subtraction` sebuah matriks 2 dimensi dengan tinggi dan lebar sama dengan `gray_img` yang mana tiap *cell* sama dengan 0.
5. Baris kosong.
6. Perulangan `y` sebanyak `h`.
7. Perulangan `x` sebanyak `w`.

8. Proses pengurangan nilai parameter *gray_img(y,x)* dengan nilai parameter *gray_background(y,x)* yang di absolutkan dan disimpan ke variabel *subtraction*.
9. Jika *subtraction(y,x)* lebih besar *T*, *subtraction(y,x)* bernilai 255. Jika kurang dari *T*, *subtraction(y,x)* bernilai 0.
10. Baris kosong.
11. Mengembalikan nilai *subtraction*.

4.8.3 Implementasi Proses *Object Filtering*

1. *Erosion*

```
1   def erosion(subtraction):
2       erosion = np.asarray(np.zeros_like(gray_img),
    dtype="uint8")
3       pixelx = [0, -1, 0, 1, 1, 0, 0, -1, -1]
4       pixely = [0, 0, -1, 0, 0, 1, 1, 0, 0]
5       h = subtraction.shape[0]
6       w = subtraction.shape[1]
7
8       for y in range(1, h - 1):
9           for x in range(1, w-1):
10              for i in range(9):
11                  py = y + pixely[i]
12                  px = x + pixelx[i]
13
14                  if subtraction[py, px] == 0:
15                      erosion[y, x] = 0
16                      break
17
18                  if i == 8:
19                      erosion[y, x] = 255
20
21      return erosion
```

Kode Program 4.3 Proses *Erosion*

Penjelasan Kode Program 4.3, pada baris ke:

1. *Method* *erosion* dengan parameter *subtraction* citra dari hasil proses *Background Subtraction*.
2. Inisialisasi variabel *erosion* sebuah matriks 2 dimensi dengan tinggi dan lebar sama dengan *gray_img* yang mana tiap *cell* sama dengan 0.
3. Inisialisasi matriks variabel *pixelx* dengan nilai 0, -1, 0, 1, 1, 0, 0, -1 dan -1.
4. Inisialisasi matriks variabel *pixely* dengan nilai 0, 0, -1, 0, 0, 1, 1, 0 dan 0.
5. Inisialisasi variabel *h* bernilai tinggi dari matriks *subtraction*.
6. Inisialisasi variabel *w* bernilai lebar dari matriks *subtraction*.
7. Baris kosong.
8. Perulangan *y* sebanyak *h*-1.
9. Perulangan *x* sebanyak *w*-1.
10. Perulangan *i* sebanyak 9.

11. Proses penambahan nilai *y* dengan nilai *pixely(y,x)* dan disimpan ke variabel *py*.
12. Proses penambahan nilai *x* dengan nilai *pixelx(y,x)* dan disimpan ke variabel *px*.
13. Baris kosong.
14. Jika *subtraction(py, px)* sama dengan 0.
15. Jika Ya, *erosion(y,x)* sama dengan 0.
16. *break*.
17. Baris kosong.
18. Jika *i* sama dengan 8.
19. Jika Ya, *erosion(y,x)* sama dengan 255.
20. Baris kosong.
21. Mengembalikan nilai *erosion*.

2. *Dilation*

```
1   def dilation(subtraction):
2       dilation = np.asarray(np.zeros_like(gray_img),
    dtype="uint8")
3       pixelx = [0, -1, 0, 1, 1, 0, 0, -1, -1]
4       pixely = [0, 0, -1, 0, 0, 1, 1, 0, 0]
5       h = subtraction.shape[0]
6       w = subtraction.shape[1]
7
8       for y in range(1, h - 1):
9           for x in range(1, w-1):
10              for i in range(9):
11                  py = y + pixely[i]
12                  px = x + pixelx[i]
13
14                  if subtraction[py, px] == 255:
15                      dilation[y, x] = 255
16                      break
17
18                  if i == 8:
19                      dilation[y, x] = 0
20
21      return dilation
```

Kode Program 4.4 Proses *Dilation*

Penjelasan Kode Program 4.4, pada baris ke:

1. *Method* *dilation* dengan parameter *subtraction* citra dari hasil proses *erosion*.
2. Inisialisasi variabel *dilation* sebuah matriks 2 dimensi dengan tinggi dan lebar sama dengan *gray_img* yang mana tiap *cell* sama dengan 0.
3. Inisialisasi matriks variabel *pixelx* dengan nilai 0, -1, 0, 1, 1, 0, 0, -1 dan -1.
4. Inisialisasi matriks variabel *pixely* dengan nilai 0, 0, -1, 0, 0, 1, 1, 0 dan 0.
5. Inisialisasi variabel *h* bernilai tinggi dari matriks *subtraction*.
6. Inisialisasi variabel *w* bernilai lebar dari matriks *subtraction*.

7. Baris kosong.
8. Perulangan *y* sebanyak h-1.
9. Perulangan *x* sebanyak *w-1*.
10. Perulangan *i* sebanyak 9.
11. Proses penambahan nilai *y* dengan nilai *pixely(y,x)* dan disimpan ke variabel *py*.
12. Proses penambahan nilai *x* dengan nilai *pixelx(y,x)* dan disimpan ke variabel *px*.
13. Baris kosong.
14. Jika *subtraction(py, px)* sama dengan 0.
15. Jika Ya, *dilation(y,x)* sama dengan 0.
16. *break*.
17. Baris kosong.
18. Jika *i* sama dengan 8.
19. Jika Ya, *dilation(y,x)* sama dengan 255.
20. Baris kosong.
21. Mengembalikan nilai *dilation*.

4.8.4 Implementasi Proses *Blob Detection*

```
1   def set_blob(py, px, obj):
2       pixelx = [0, -1, 0, 1, 1, 0, 0, -1, -1]
3       pixely = [0, 0, -1, 0, 0, 1, 1, 0, 0]
4
5       for i in range(9):
6           py = py + pixely[i]
7           px = px + pixelx[i]
8
9           if thresh[py, px] == 255:
10              thresh[py, px] = obj
11              set_blob(py, px, obj)
12
13          if i == 8:
14              np.savetxt(str(obj) + ".csv", thresh,
    delimiter=",")
```

Kode Program 4.5 Proses *Blob Detection*

Penjelasan Kode Program 4.5, pada baris ke:

1. *Method* *set_blob* dengan parameter *py* piksel koordinat *y*, *px* piksel koordinat *x* dan *obj* sebagai nilai unik pembeda tiap objek.
2. Inisialisasi matriks variabel *pixelx* dengan nilai 0, -1, 0, 1, 1, 0, 0, -1 dan -1.
3. Inisialisasi matriks variabel *pixely* dengan nilai 0, 0, -1, 0, 0, 1, 1, 0 dan 0.
4. Baris kosong.
5. Perulangan *i* sebanyak 9.
6. Proses penambahan nilai *y* dengan nilai *pixely(y,x)* dan disimpan ke variabel *py*.
7. Proses penambahan nilai *x* dengan nilai *pixelx(y,x)* dan disimpan ke variabel *px*.

8. Baris kosong.
9. Jika *thresh(py, px)* sama dengan 255.
10. Jika Ya, *thresh(y,x)* sama dengan *obj*.
11. Baris kosong.
12. Jika *i* sama dengan 8.
13. Jika Ya, simpan hasil *thresh* ke dalam *file csv*.

4.8.5 Implementasi Proses Deteksi Objek

1. *Blob Contour*

```
1   def blob_contour(obj):
2       h = thresh.shape[0]
3       w = thresh.shape[1]
4       contour = []
5       minx = 1
6       miny = 1
7       maxx = 1
8       maxy = 1
9
10      for y in range(0, h):
11          for x in range(0, w):
12              if thresh[y, x] == obj:
13                  if minx == 1 and miny == 1:
14                      minx = x
15                      miny = y
16                      maxx = x
17                      maxy = y
18                  else:
19                      if minx > x:
20                          minx = x
21                      if maxx < x:
22                          maxx = x
23
24                      if miny > y:
25                          miny = y
26                      if maxy < y:
27                          maxy = y
28
29      contour = [minx, miny, maxx, maxy]
30
31      return contour
```

Kode Program 4.6 Proses *Blob Contour*

Penjelasan Kode Program 4.6, pada baris ke:

1. *Method* *blob_contour* dengan parameter *obj* sebagai nilai unik pembeda tiap objek.
2. Inisialisasi variabel *h* bernilai tinggi dari matriks *thresh.*
3. Inisialisasi variabel *w* bernilai lebar dari matriks *thresh.*
4. Inisialisasi variabel *contour* sebuah matriks kosong.
5. Inisialisasi variabel *minx* *bernilai 1.*
6. Inisialisasi variabel *miny* *bernilai 1.*
7. Inisialisasi variabel *maxx* *bernilai 1.*
8. Inisialisasi variabel *maxy* *bernilai 1*

9. Baris kosong.
10. Perulangan *y* sebanyak *h*.
11. Perulangan *x* sebanyak *w*.
12. Jika *thresh(py, px)* sama dengan *obj*.
13. Jika *minx* sama dengan 1 dan *miny sama dengan 1*
14. Jika Ya, *minx* sama dengan *x*.
15. Jika Ya, *miny* sama dengan *y*.
16. Jika Ya, *maxx* sama dengan *x*.
17. Jika Ya, *maxy* sama dengan *y*.
18. Baris kosong.
19. Jika tidak.
20. Jika *minx* lebih besar dari *x*.
21. Jika Ya, *minx* sama dengan *x*.
22. Jika *maxx* kurang dari dari *x*.
23. Jika Ya, *maxx* sama dengan *x*.
24. Baris kosong.
25. Jika *minx* lebih besar dari *x*.
26. Jika Ya, *minx* sama dengan *x*.
27. Jika *maxx* kurang dari dari *x*.
28. Jika Ya, *maxx* sama dengan *x*.
29. Baris kosong.
30. Simpan nilai *minx*, *miny*, *maxx* dan *maxy* ke dalam matriks variabel *contour*.
31. Baris kosong.
32. Mengembalikan nilai *contour*.

2. *Blob Contours*

```
1   def blob_contours(obj):
2       h = thresh.shape[0]
3       w = thresh.shape[1]
4
5       for y in range(0, h):
6           for x in range(0, w):
7               if thresh[y, x] == 255:
8                   set_blob(y, x, obj)
9                   contour = blob_contour(obj)
10                  contours.append(contour)
11                  obj = obj + 1
12                  blob_contours(obj)
```

Kode Program 4.7 Proses *Blob Contours*

Penjelasan Kode Program 4.7, pada baris ke:

1. *Method* *blob_contour* dengan parameter *obj* sebagai nilai unik pembeda tiap objek.
2. Inisialisasi variabel *h* bernilai tinggi dari matriks *thresh*.
3. Inisialisasi variabel *w* bernilai lebar dari matriks *thresh*.
4. Baris kosong.
5. Perulangan *y* sebanyak *h*.

6. Perulangan *x* sebanyak *w.*
7. Jika *thresh(y,x)* sama dengan *255*.
8. Memanggil *method* `set_blob(y, x, obj).`
9. Memanggil *method* *blob_contour(obj)* dan hasilnya disimpan ke variabel *contour.*
10. Setiap *contour* ditambahkan ke dalam matriks *contours.*
11. *Obj* sama dengan *obj* ditambah dengan 1.
12. Memanggil *method* *blob_contours(obj)*.

BAB 5 PENGUJIAN DAN ANALISIS

Pada bagian ini dibahas tekait proses pengujian terhadap sistem yang telah diimplementasikan. Pada proses pengujian dilakukan berdasarkan pada perancangan pengujian yang telah dirancang sebelumnya. Pengujian sistem meliputi pengujian pengaruh proses filter dan pengaruh *threshold*. Dalam semua kondisi setiap pengujian dilakukan sebanyak 10 citra berwarna setiap *video* dan 1 citra *background*. Setiap citra berukuran 426x240 piksel, kemudian diambil nilai rata-rata evaluasinya, evaluasi sistem menggunakan *error* dan akurasi. Dilakukannya proses pengujian untuk mengetahui hasil evaluasi sistem terhadap implementasi metode yang telah digunakan.

5.1 Pengujian dan Analisis Pengaruh Proses Filter

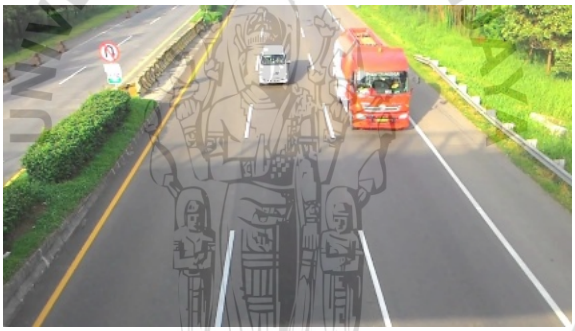
Dalam pengujian proses filter terdapat 2 parameter uji yaitu *kernel* dan iterasi. Parameter *kernel* menggunakan *kernel* 3x3 dan *kernel* 5x5, sedangkan untuk parameter iterasi menggunakan kombinasi 3, 4 dan 5 untuk proses *erosion* dan *dilation*. *Threshold* yang digunakan untuk pengujian ini adalah 30. Pengujian ini bertujuan untuk mengetahui *kernel* yang cocok agar bisa diimplementasikan oleh sistem dan kombinasi yang sesuai dengan *dataset* yang sudah tersedia. Pada pengujian ini dilakukan perhitungan *error* dengan menggunakan evaluasi *Mean Square Error* dengan mengurangi jumlah yang dihasilkan oleh sistem dengan jumlah sebenarnya yang dikuadratkan dan dibagi dengan jumlah pengujian. Kombinasi terbaik jika *error* mempunyai nilai terendah dari kombinasi yang lain dengan perhitungan seperti pada Persamaan 2.4 yang dilakukan pada setiap kombinasi pengujian. Untuk nilai hasil evaluasi pengujian pengaruh proses filter dapat dilihat pada Tabel 5.1.

Tabel 5.1 Hasil Pengujian Pengaruh Proses Filter

Pengujian pada percobaan ke-				**1**	**2**	**3**	**..**	**..**	**..**	**18**	**19**	**20**	**MSE**
Jumlah sebenarnya				**3**	**2**	**3**	**..**	**..**	**..**	**5**	**5**	**5**	
No	***kernel***	**Iterasi *erosion***	**Iterasi *dilation***	**Jumlah yang diperoleh sistem**									
1	3x3	3	3	9	2	6	..	..	..	10	9	9	**40,8**
2	3x3	3	4	9	2	4	..	..	..	7	8	7	**16,15**
3	3x3	3	5	6	2	3	..	..	..	5	5	4	**4,95**
4	3x3	4	3	5	3	4	..	..	..	6	7	6	**7,3**
5	3x3	4	4	5	3	4	..	..	..	6	6	6	**6,3**
6	3x3	4	5	5	2	4	..	..	..	4	5	6	**4,75**
7	3x3	5	3	4	1	4	..	..	..	4	4	3	**3,85**
8	3x3	5	4	4	1	4	..	..	..	4	4	3	**3,65**
9	3x3	5	5	4	1	4	..	..	..	4	4	3	**3,5**
10	5x5	3	3	1	1	4	..	..	..	4	3	0	**5,4**
11	5x5	3	4	1	1	4	..	..	..	3	2	0	**6,25**

12	5x5	3	5	1	1	4	..	..	..	3	2	0	**7,45**
13	5x5	4	3	1	0	2	..	..	..	3	3	0	**8,6**
14	5x5	4	4	1	0	2	..	..	..	3	3	0	**8,6**
15	5x5	4	5	1	0	2	..	..	..	2	2	0	**9,5**
16	5x5	5	3	1	0	0	..	..	..	2	1	0	**11,3**
17	5x5	5	4	1	0	0	..	..	..	2	1	0	**11,3**
18	5x5	5	5	1	0	0	..	..	..	2	1	0	**11,3**

Hasil *error* terendah sebesar 3,5 dengan *kernel* 3x3, *erosion* sebanyak 5 iterasi dan *dilation* sebanyak 5 iterasi dan *error* terbesar adalah 40,8 karena hasil dari iterasi *erosion* yang terlalu sedikit belum benar-benar menghilangkan derau dari hasil proses pemisahan objek dengan *background.* Dalam pengujian ini untuk *kernel* 3x3 *erosion* dan *dilation* terkecil ketika terdapat pada *kernel* 3x3, *erosion* dan *dilation* terkecil yaitu 3 kali iterasi. Mengalami perbaikan *error* jika nilai *dilation* dinaikkan sehingga mendapatkan *error* terkecil pada baris berwarna hijau dan *error* terbesar pada baris berwarna kuning seperti Tabel 5.1. Tabel lebih lengkap dapat dilihat pada Lampiran 1.



Gambar 5.1 Citra Pengujian Proses Filter Ke-7

Gambar 5.2 Hasil Ke-7 *Error* Terkecil

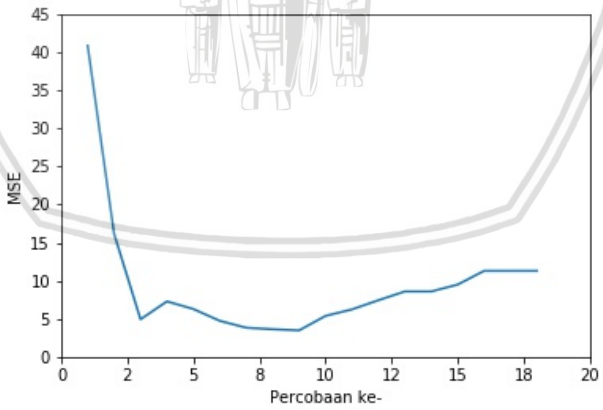
Gambar 5.3 Hasil Ke-7 *Error* Terbesar

Gambar 5.4 Citra Pengujian Proses FIlter Ke-17

Gambar 5.6 Hasil Ke-17 *Error* Terkecil

Gambar 5.5 Hasil Ke-17 *Error* Terbesar

Hasil pengujian proses filter ini diambil sebuah citra asli dan citra hasil dari *error* terendah dan terbesar. Seperti pada Gambar 5.2 menghasilkan 2 jumlah objek sesuai dengan banyak objek sebenarnya Gambar 5.1. Sedangkan pada Gambar 5.2 terdapat sebuah objek yang terdeteksi oleh sistem dianggap lebih dari 1 objek.



Gambar 5.7 Grafik Pengujian Proses Filter

Berdasarkan grafik pada Gambar 5.7, terlihat bahwa hasil evaluasi sistem pada pengujian ini berbeda nilai *error* ketika nilai kombinasi *kernel*, *erosion* dan *dilation* berbeda-beda. Kombinasi *error* terendah ketika berada pada percobaan ke-9 *kernel* 3x3, *erosion* sebanyak 5 iterasi dan *dilation* sebanyak 5 iterasi.

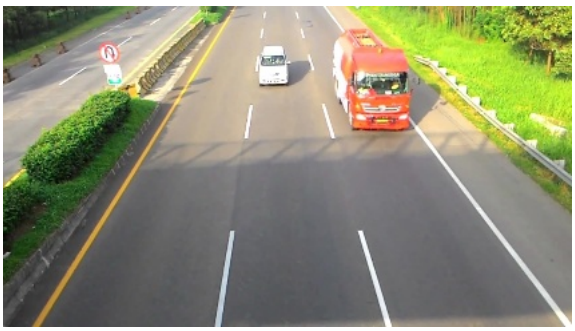
5.2 Pengujian dan Analisis Pengaruh Jumlah *Threshold*

Pengujian ini berguna untuk mengetahui pengaruh *threshold* terhadap hasil evaluasi sistem yang telah dibuat. Dalam pengujian ini dilakukan evaluasi dengan jumlah *threshold* 10, 20, 30, 40 dan 50. Nilai dari pengujian ini digunakan untuk mengetahui perubahan yang signifikan dan bertujuan untuk mendapatkan evaluasi terkecil dan evaluasi tertinggi. Pada pengujian sebelumnya parameter uji digunakan dalam pengujian ini dengan *error* terendah yaitu, *kernel* 3x3, *erosion* sebanyak 5 iterasi dan *dilation* sebanyak 5 iterasi. Dalam pengujian ini didapatkan nilai evaluasi error dan akurasi yang berbeda-beda. Untuk nilai hasil evaluasi pengujian pengaruh jumlah *threshold* dapat dilihat pada Tabel 5.2. Tabel lebih lengkap dapat dilihat pada Lampiran 2.

Tabel 5.2 Hasil Pengujian Pengaruh Jumlah *Threshold*

Pengujian pada percobaan ke-				**1**	**2**	**3**	**..**	**..**	**..**	**18**	**19**	**20**	**MSE**
Jumlah sebenarnya				**3**	**2**	**3**	**..**	**..**	**..**	**5**	**5**	**5**	
No	**Threshold**	**Iterasi *erosion***	**Iterasi *dilation***	**Jumlah yang diperoleh sistem**									
1	10	5	5	7	1	5	..	..	..	5	8	11	**16,5**
2	15	5	5	6	1	4	..	..	..	3	4	8	**5,5**
3	20	5	5	4	1	4	..	..	..	3	4	5	**6,2**
4	25	5	5	4	1	4	..	..	..	3	4	5	**4,55**
5	30	5	5	4	1	4	..	..	..	4	4	3	**3,6**
6	35	5	5	2	1	4	..	..	..	4	4	1	**5**
7	40	5	5	2	1	3	..	..	..	4	5	0	**5,4**
8	45	5	5	2	1	2	..	..	..	4	5	0	**5,25**
9	50	5	5	1	1	2	..	..	..	4	5	0	**5,95**

Hasil *error* terendah sebesar 3,6 dengan nilai *threshold* adalah 30 dan *error* terbesar sebesar 16,5 dengan nilai *threshold* adalah 10. Dalam pengujian ini *error* mengalami penurunan dengan batas maksimal *threshold* bernilai 30 dan mengalami penambahan ketika *threshold* bernilai 40 sampai kelipatan 5, sehingga mendapatkan *error* terkecil pada baris berwarna hijau dan *error* terbesar pada baris berwarna kuning seperti Tabel 5.2.



Gambar 5.8 Citra Pengujian *Threshold* Ke-7

Gambar 5.10 Hasil Ke-7 *Error* Terkecil

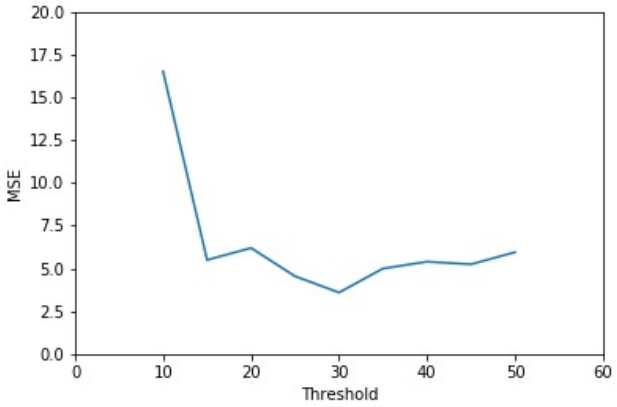
Gambar 5.9 Hasil Ke-7 *Error* Terbesar

Gambar 5.11 Contoh Citra Pengujian *Threshold* Ke-17

Gambar 5.13 Hasil Ke-17 *Error* Terkecil

Gambar 5.12 Hasil Ke-17 *Error* Terbesar

Hasil pengujian *threshold* ini diambil sebuah citra asli dan citra hasil dari *error* terendah dan terbesar. Seperti pada Gambar 5.10 menghasilkan 3 jumlah objek sesuai dengan banyak objek sebenarnya Gambar 5.11. Sedangkan pada Gambar 5.12 terdapat sebuah objek yang terdeteksi oleh sistem dianggap lebih dari 1 objek.



Gambar 5.14 Grafik Pengujian Jumlah *Threshold*

Berdasarkan grafik pada Gambar 5.14, terlihat bahwa hasil evaluasi sistem pada pengujian ini berbeda nilai *error* ketika nilai *threshold* bervariasi. Hasil *error* terendah ketika *threshold* bernilai 30.

5.3 Pengujian Evaluasi Kinerja

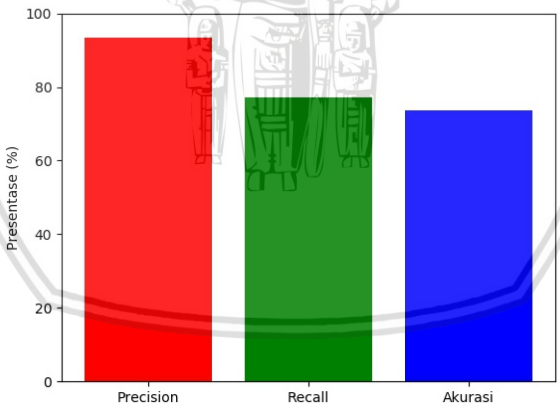
Pengujian ini berguna untuk mengevaluasi kinerja *precision*, *recall* dan akurasi pada penggabungan metode antara *Blob Detection* dan *Background Subtraction* yang dihitung berdasarkan empat nilai, yaitu *True Positive* (TP), *False Positive* (FP), *False Negative* (FN) dan *True Negative* (TN).

Dari hasil percobaan dengan 20 *dataset*, diperoleh tingkatan probabilitas seperti disajikan pada Tabel 5.3.

Tabel 5.3 Hasil Evaluasi Kinerja

Jenis Kinerja	Presentase (%)
Precision	93,44
Recall	77,03
Akurasi	73,75

Dari hasil perhitungan diketahui bahwa *precision* sebesar 93,44%, *recall* sebesar 77,03% dan akurasi sebesar 73,75%. Maka nilai *precision*, *recall* dan akurasi perlu ditingkatkan lagi dengan menambahkan parameter pengujian dan memperbanyak *dataset* dengan kondisi yang berbeda-beda. Adapun hasil evaluasi kinerja digambarkan melalui grafik pada Gambar 5.15.



Gambar 5.15 Evaluasi Kinerja

Berdasarkan Gambar 5.15 dapat dikatakan bahwa metode *Blob Detection* ini dapat digunakan sebagai metode pendeteksian kendaraan pada jalan raya. Namun, untuk letak kendaraan yang berdekatan pada citra, metode *Blob Detection* mendeteksi dua kendaraan yang berdekatan menjadi satu objek, sehingga akurasi nilai kepadatan berkurang. Hal ini dikarenakan metode *Blob Detection* mendeteksi suatu objek berdasarkan citra biner yang dihasilkan. Oleh karena itu, pada penelitian selanjutnya, perlu dilakukan pemisahan objek kendaraan yang berdekatan, sehingga meningkatkan nilai akurasi pendeteksian. Dari hasil eksperimen dapat disimpulkan bahwa *precision* sebesar 93,44%, *recall*

sebesar 77,03% dan tingkat akurasi yang terbilang rendah, yaitu sebesar 73,75%. Namun, hal ini masih dikatakan efektif karena mencapai nilai di atas 50%.

BAB 6 PENUTUP

Bagian ini memuat kesimpulan dan saran terhadap skripsi. Kesimpulan dan saran disajikan secara terpisah, dengan penjelasan sebagai berikut:

6.1 Kesimpulan

Berdasarkan hasil penelitian ini, dapat disimpulkan beberapa hal mengenai penentuan jumlah kendaraan menggunakan *Blob Detection* dan *Background Subtraction*.

1. Penentuan jumlah kendaran dapat dilakukan menggunakan algoritme *Blob Detection* dengan menggunakan algoritme *Background Subtraction* pada proses segmentasi untuk penentuan jumlah kendaran dalam sebuah citra. Pada sistem ini perlu dilakukan evaluasi proses filter dan *threshold* untuk mendapatkan hasil evaluasi terbaik. Setelah dilakukan evaluasi, sistem dapat menentukan jumlah kendaraan dalam sebuah citra dengan hasil yang optimal.
2. Kinerja menghasilkan tingkat akurasi penggunaan *Blob Detection* dan *Background Subtraction* pada kondisi lengang lebih baik daripada kondisi ramai sebesar 73,75%. Kesalahan pada kondisi ramai lebih banyak disebabkan karena posisinya yang bergerombol dan berdekatan satu sama lain, sehingga sistem membaca banyak kendaraan tersebut sebagai satu kesatuan objek kendaraan dengan dimensi besar.

6.2 Saran

Berdasarkan hasil penelitian ini, beberapa saran mengenai penentuan jumlah kendaraan menggunakan *Blob Detection* dan *Background Subtraction*.

1. Melakukan pengujian pada setiap parameter dari model untuk menentukan parameter seperti *threshold* dan kombinasi proses filter.
2. Dapat diimplementasikan ke dalam bentuk *video* dan digunakan secara *realtime* dengan mengoptimasi setiap langkah-langkah proses.

DAFTAR PUSTAKA

Adinugroho, S. & Sari, Y. A., 2018. *Implementasi Data Mining Menggunakan Weka.* s.l.:UB Press.

Cahyana, F. M., 2014. Perancangan Program Penghitung Jumlah Kendaraan Di Lintasan Jalan Raya Satu Arah Menggunakan Bahasa Pemrograman C++ Dengan Pustaka OpenCV 2.4.4. *Jurnal Mahasiswa TEUB,* Volume 2.

H, A. N., Ichwan, M. & Putra, . I. M. S., 2015. Segmentasi Citra Untuk Deteksi Objek Warna Pada Aplikasi Pengambilan Bentuk Citra Rectangle. *Itenas Library.*

Harwendhani, I. C., 2016. Sistem Pendeteksi Jumlah Mobil Dalam Intelligent Transport System (ITS) Menggunakan Metode Viola-Jones. *Semantik Informatika,* Volume 2.

Hidayati, Q., 2017. Kendali Lampu Lalu Lintas dengan Deteksi Kendaraan Menggunakan Metode Blob Detection. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI) ,* Volume 6.

Kurniawan, W. R., 2015. Purwapura Sistem Klasifikasi dan Penghitung Jumlah Kendaraan Bermotor Menggunakan Kamera Webcam Berbasis Citra Digital Di PT. Industri Telekomunikasi Indonesia (Persero). *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI).*

Mediawati, M. G. S. K., 2017. Stres Pengendara Motor pada Kemacetan Lalulintas di Kota Semarang. *Digilib Unnes.*

Menegaz , G., 2017. *Morphological Image Processing.* Verona: Verona University.

Pitoko, R. A., 2017. *Macet Jakarta Terburuk Ke-4 di Dunia.* [Online] Available at: https://properti.kompas.com/read/2017/02/22/215638721/macet.jakarta.terburuk.ke-4.di.dunia
[Accessed Februari 2018].

Qi, X., Li, X. & Zhang, H., 2013. Research of Paper Surface Defects Detection System Based on Blob Algorithm. *IEEE Xplore,* Agustus.

Samir, M. I., Zuraiyah, T. A. & Aryani, A. S., 2016. Penerapan Algoritma Background Substraction Untuk Tracking dan Klasifikasi Kendaraan. *Journal Of Mechanical Engineering And Mechatronics,* Volume 1.

Wang, Z. & Bovik, A. C., 2009. Mean Squared Error: Love It or Leave It?. *IEEE Xplore,* Januari, 26(1), pp. 98 - 117.

Wirayuda, T. A. B., 2006. Pemanfaatan Operasi Morphologi Untuk Proses Pendeteksian Sisi Pada Pengolahan Citra Digital. *Seminar Nasional Informatika (SEMNASIF),* Volume 1.